

August 1988
7 DM · 60 öS · 7 sFr

Die Mikrocomputer-Zeitschrift

8/88

Generic CADD:

CAD für Profis

**Hercules-Karte:
Grafik-Bibliothek**

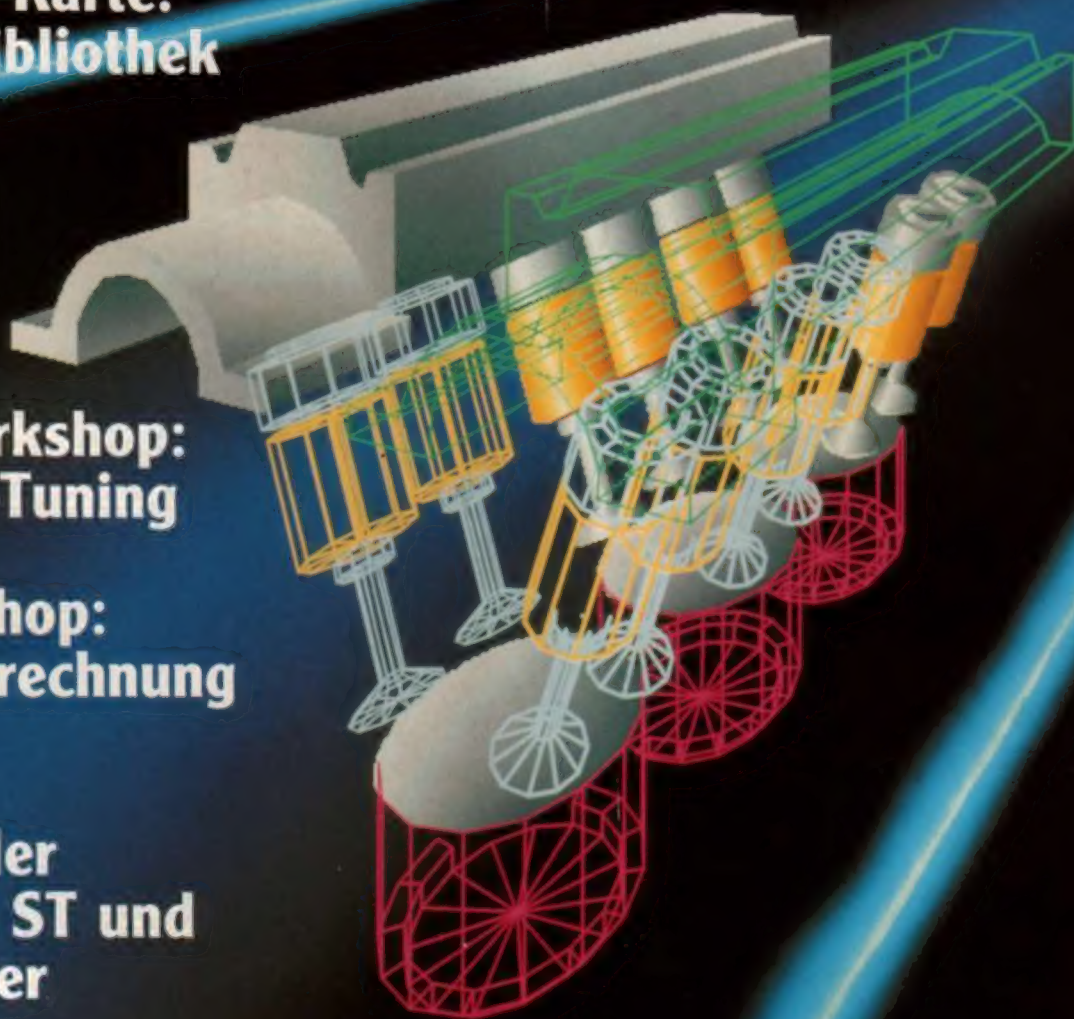
Trend: Low-cost-386
von Compaq
IBM-Modell 70

**DOS-Workshop:
RS232C-Tuning**

**C-Workshop:
Matrizenrechnung**

**Test:
C-Compiler
für Atari ST und
Transputer**

**GFA-Basic 3.0
Schneider EGA-AT
Manuscript
Norton Guides**



NEUGIERIG AUF PLANTRON



PLANTRON

Computer Vertriebsgesellschaft mbH

Höhestraße 28 · D-6380 Bad Homburg v.d.H.

Telefon (06172) 81031* · Telefax (06172) 81036

Telex 417 410 placo d

EDITORIAL

CAD auf PC, das ist bisher von den Profis immer belächelt worden. Mit dem Aufkommen der Rechner der AT-Klasse und dem Erscheinen der 386er auf dem Markt nehmen aber auch die Profis mehr und mehr zur Kenntnis, daß man mit den PC rechnen muß. In dieser Ausgabe stellen wir das Programmsystem Generic CADD vor, das erstaunliche Leistungsdaten aufweist. Darüber hinaus schildert Erich Jäger die Möglichkeiten, die er mit einem PC und geeigneter Software besitzt, um Computergrafik zu betreiben. Mittlerweile stammt eine ganze Reihe von mc-Titelbildern aus seinem Computer. Dabei erweist sich die „Einheit zwischen Werkzeug und Thema“ – auf einem Mikrocomputersystem wird das Titelbild von mc, der Mikrocomputerzeitschrift entworfen – als fruchtbar und gibt dem Ergebnis einen Reiz, wie wir hoffen.

Liebe Leser,

Es wird ein heißer Computerherbst. IBM und Compaq stellten je zwei neue PC vor. IBM die Modelle 70 und 50-Z, Compaq zwei 386er. Von Mitac aus Taiwan kommt ein 286er mit 386er-Geschwindigkeit, von Kaypro ein AT. Der Trend ist ganz klar, Programmpakete wie Generic CADD, AutoCAD und andere benötigen entsprechend leistungsfähige Rechner. Die PC mauern sich mehr und mehr zu preiswerten Workstations. Das, was bisher größeren Firmen vorbehalten war, kann jetzt auch auf dem Schreibtisch eines Normalbürgers stehen und dessen Leistungen besser zur Geltung bringen. In diesem Trend liegt auch Apple, dessen Macintosh II zwar als hochpreisig gilt, gemessen an seiner Leistungskraft aber gut im Rennen liegt. Hier fehlt höchstens noch einiges an öffentlichen Informationen über Programmpakete und Systemverhalten. Soweit generell einige Beispiele zum Trend. In dieser Ausgabe gibt es orientierende Berichte über die IBMs und die Compaqs im aktuellen Teil. Schritt für Schritt werden wir für Sie ausführliche Tests durchführen.

mc ist nicht nur Neuigkeiten verpflichtet. Mit mc sollen Sie – falls das überhaupt nötig sein sollte – etwas lernen und vor allem auch Arbeitserleichterung bekommen. In dieser Ausgabe bringen



wir ein C-Programm-Paket, das die Bearbeitung von Matrizen beliebiger Größe recht komfortabel gestaltet – einfach einbinden und compilieren. Und ganz nebenbei erfahren Sie in diesem Beitrag Wesentliches über den Aufbau von C, Fortran und die kleinen (aber wesentlichen!) Unterschiede zwischen beiden. Außerdem berichten wir über C-Compiler für den Atari ST und für Transputer.

Peter Norton, das ist ein berühmter Name, wenn es um das Testen und Programmieren von PC geht. Von ihm kommen jetzt die Norton Guides, ein Informationssystem, das es in sich hat. Kurz gesagt, man packt sich mit den Norton Guides die Handbuchinformationen zu einem bestimmten Thema in den Hintergrund auf dem PC und kann dann bei Bedarf während der Arbeit diese Informationen abrufen. Weshalb das so nützlich ist? Alle, die viel mit einem PC arbeiten und alle möglichen Programmiersprachen benutzen, von Assembler bis C, werden es mit Sicherheit begrüßen, wenn die Frage: „Wie war das gleich noch mal in Turbo-C (Pascal, Microsoft-C, ...)?“ schnell und direkt gleich auf dem Computer geklärt werden kann – ohne langes Blättern im entsprechenden Handbuch, das vielleicht im Moment nicht greifbar ist. Es könnte für Sie viel Arbeitserleichterung bedeuten, wenn Sie unseren Test auf Seite 50 beachten.

Ulrich Rohde

DIE UNIC-ICB-IDEE

XT/AT-KOMPATIBLE
COMPUTER IN
INDUSTRIELLEM DESIGN

Die starke Verbreitung von IBM-kompatiblen Rechnersystemen hat es mit sich gebracht, daß eine sehr große Anzahl von entsprechenden Betriebssystemen, Programmiersprachen, Utilities und Programmgeneratoren rasch und preisgünstig erhältlich ist. Die Verwendung dieser Programme für den industriellen Einsatz (Maschinensteuerung, Betriebsdatenerfassung) scheiterte meist daran, daß normale Rechnersysteme für raue Umgebungsbedingungen nicht ausgelegt waren (erschütterungsempfindlich, Probleme mit Staub, Temperatur und Feuchtigkeit).



VORTEILE DER UNIC-ICB-RECHNER

100 % softwarekompatibel zu IBM XT/AT-Rechnern. Vollständiger Aufbau auf Doppel- und Einfach-Europakarten – genormte Gehäuse, beliebige Schutzarten. Erweiterter Temperaturbereich durch entsprechende Steckverbindungen und Bauteilauswahl. Erschütterungsempfindlich durch Einsatz nichtrotierender Massenspeicher (RAM/EPROM-Floppy).

Vollständig modular aufgebautes System, dessen Komponenten für Softwareentwicklung und eventuell Debugging auch am Personalcomputer angeschlossen werden können. Original IBM-Erweiterungskarten (Netzwerkarten, verschiedene Schnittstellenkarten usw.) sind über eine Adapterkarte direkt auf ICB-Systemen einsetzbar.

Durch die Verwendungsmöglichkeit eines Personalcomputers zur Programmentwicklung für eine industrielle Applikation entfällt die ansonsten notwendige Anschaffung eines eigenen Entwicklungssystems (VME, ECB usw.).

Durch konsequente 19"-Technik (zum Einsatz kommen Platinen im Einfach- und/oder Doppel-Euro-Format) sind ohne großen Aufwand beliebige Schutzarten nach DIN 40 050 realisierbar.

indutronic

Dipl.-Ing. Kreiger GmbH & Co. KG
D-8011 Pöding/München
Gruberstraße 46
Telefon (0 81 21) 7 01-0
Telefax (0 81 21) 7 13 64

Ich bitte um Zusendung weiterer Informationen:

Name:

Firma:

Straße:

Plz./Ort:

Telefon:

INHALT

INFO

OSF entwickelt Software-Standards	9
Omkron-Basic: Standard für alle Atari ST	23
Laser-Drucker mit Hyperscript-Prozessoren	24
GEM/3 verfügbar	30
PC-Markt in Deutschland	33
CAD400 von Dataport auf IBM 6150	107
Minolta mit Chipkarten	113

TEST

CAD für schnelle AT-Rechner	40
Mit dem CAD-Programm Generic CADD kommt man auf einem gut ausgestatteten AT schnell zu ansprechenden Grafiken.	
Pascal-SC: Mathematik auf dem Atari ST	46
Ein C-Compiler, der für die Lösung wissenschaftlicher Aufgaben prädestiniert ist.	
GFA-Basic 3.0 für Atari ST	47
Die neue Version des Basic-Interpreters hat nur noch wenig mit Standard-Basic zu tun.	
Die Norton-Guides	50
Ein Datenbank-Programm mit umfangreichen Handbuch-Datenbanken vereinfacht die Programmierung am PC.	
Turbo-C für den Atari ST	52
Der Renner für PC jetzt auch auf Atari ST.	
C mit Lichtgeschwindigkeit	54
Die neueste Version des Megamax-C für den Atari ST.	
Erster Eindruck: Mitac 2000VE	57
Fast so schnell wie ein 80386-PC ist der preiswerte AT von Mitac.	
Textverarbeitung für den speziellen Einsatz	58
Für sehr lange Dokumente gedacht ist Manuscript von Lotus.	
Angenehmes Turbo-Debugging	61
Symbolischer Debugger für Turbo-Pascal 4.0	
Ein schnelles Schneiderlein	62
Der EGA-AT ist sehr klein und ganz schön schnell.	
Ein C-Compiler für die mc-Transputerkarte	64
Da freuen sich alle Transputer-Freunde: Der C-Compiler für die mc-Transputerkarte entspricht dem Kernighan-/Ritchie-Standard.	
Kompatibel, aber schnell	129
Die Datenbank-Programmiersprache dBase hat sich weitgehend durchgesetzt. Ein kompatibles Produkt, das schneller und preiswerter als das Original ist, macht neugierig.	

DOS-WORKSHOP

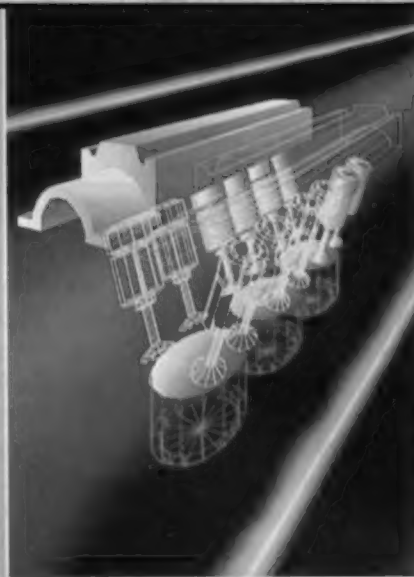
Schnelle serielle Schnittstelle	68
Wer wissen möchte, wie mit der seriellen Schnittstelle Übertragungsgeschwindigkeiten von 115 200 Baud erreicht werden, ist in unserem DOS-Workshop willkommen.	

C-WORKSHOP

Matrizenrechnung in C	73
Wer von Fortran nach C umsteigt oder einfach Standard-Matrizen-Funktionen in C sucht, ist zum C-Workshop eingeladen.	

CAD für Profis

An CAD-Programmen, die den Anforderungen von Technikern genügen, gibt es bereits eine reiche Auswahl. Ein Vertreter dieser Gattung, bei dem man das Rad nicht immer aufs neue erfinden muß, ist Generic CADD. Zu dem Programm gibt es viele Bibliotheken und Module, mit denen man selbst umfangreiche Zeichnungen in kürzester Zeit fertig hat. Selbst mit den ausgefallenen Grafikkarten und den exotischsten Ein- und Ausgabegeräten kommt Generic CADD klar. **Seite 40**



Computergrafik: Erich Jäger



C-Compiler für die mc-Transputerkarte

Transputer waren bislang nur in Occam oder Assembler zu programmieren. Um die weitere Entwicklung dieser innovativen Technik zu unterstützen, stellen wir einen Transputer-C-Compiler vor. Mit seiner Hilfe sind

C-Programmierer aus der Unix- oder DOS-Welt in der Lage Applikationen zu entwickeln, ohne eine neue Sprache lernen zu müssen. Gegendüber den von gebräuchlichen Occam-Compilern erzeugten Programmen sind sie mit Transputer-C kompiliert etwa um 10 Prozent schneller.

Seite 64



Der Signal-EMUF

In diesem Teil unserer Serie über den Signalprozessor TMS 32010 steht die Software im Vordergrund. Dabei gehen wir ausführlich auf den Befehlssatz des Prozessors ein und zeigen an einem Beispiel für einen digitalen Filter, wie man ihn programmiert. Außerdem geben wir ein paar Hinweise, wie man die Betriebssicherheit des Signal-EMUF erhöhen kann. Wer sich noch mehr mit dem Thema befassen möchte, findet zahlreiche Literaturhinweise.

Seite 80



Ein schnelles Schneiderlein

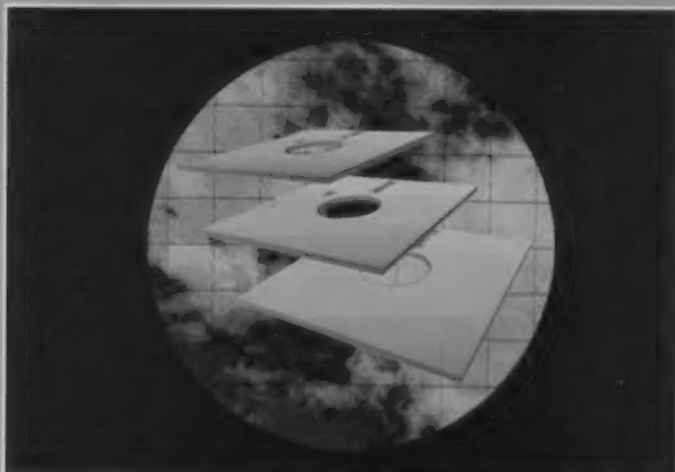
Mit dem PC 2640 bringt Schneider einen schnellen EGA-AT auf den Markt. Sein Innenleben ist ähnlich dem des mc-modular-AT; passive Bus- und CPU-Platine als Einsteckkarte zeichnen dieses Konzept aus. Alle Ein-/Ausgabe-Schnittstellen des Systems sind auf einer weiteren Steckkarte untergebracht. Das Resultat: ein schneller aber kompakter Rechner. Ausgeliefert wird der EGA-AT mit MS-DOS 3.3, GEM und 3½-Zoll-Laufwerken im PS/2-Format.

Seite 62

Neue 80386 von Compaq und IBM

Dank einer 16-Bit-Version des 80386 von Intel wird es bald zahlreiche preiswerte Super-PC geben. Den Anfang macht Compaq mit dem Deskpro 386s, der besonders kompakt aufgebaut ist. An die Grenze der derzeit mit PC erreichbaren Leistungen stoßen Compaq und IBM mit ihren 25-MHz-Modellen. Wir haben die wichtigsten Eigenschaften, Preise und die Auswirkungen auf die bestehende Produktpalette zusammengefaßt.

Seite 160



Turbo-C für den Atari ST

Den beliebten C-Compiler für PC, Turbo-C von Borland/Heimsoeth, gibt es jetzt auch für den Atari ST. Dank deutscher Entwicklungshilfe bringt das amerikanische Softwarehaus diesen Compiler jetzt auf den Markt. Es wurde vom Macintosh auf den Atari ST portiert und läuft bereits zufriedenstellend.

Seite 52

INHALT

GRUNDLAGEN

Ein Titelbild entsteht	42
LAN-Technik und uucp	78
Kommunikation von Unix-Rechnern untereinander.	
Der Signal-EMUF	80
Damit die 5-MIPS-Maschine auch bedient werden kann, bringen wir die erforderliche Software.	
Ergänzungen zum Signal-EMUF	83

SOFTWARE

Hercules-Grafik-Bibliothek	86
Ähnlich GKS ist unsere Bibliothek in Turbo-Pascal aufgebaut.	
Diskettenzugriff ohne Problem	97
Ein kleines Programm in Turbo-Pascal übernimmt die Überwachung des Diskettenschachtes.	
Maus-Funktionen für Turbo-Basic	98
Durch einige Prozeduren versteht sich der neue Basic-Compiler auch mit der Maus.	
Bäumchen wechsele dich	104
Denksportaufgabe in C gelöst.	
EGA auf 24-Nadel-Drucker	106
Um die volle Leistung eines 24-Nadel-Druckers in Kombination mit einer EGA-Karte zu erlangen, bedarf es eines geeigneten Treibers.	
Handbremse für den AT	108
Utility für PC-Anwender, die Programme langsamer macht.	
Aufs Ganze gehen	109
Normalisierte Zahlen sorgen für schnellste Berechnungen.	
Schreibschutz für die Atari-Festplatte	114
Durch ungeliebte Zeitgenossen, die sich auf das Programmieren von Virus-Programmen verlegt haben, hat dieses Programm eine große Bedeutung für alle Atari ST-Anwender.	
Typisch Prolog	116
Hier geht es um das Differenzieren und Vereinfachen von symbolischen Ausdrücken.	
Sortiertes Inhaltsverzeichnis mit Turbo 4.0	125
Weniger Arbeit dank der neuen Funktionen von Turbo-Pascal 4.0.	

HARDWARE

Neue Chips für die PC-Welt	49
Unter G-2 firmiert ein Chip-Produzent, der unter anderem wesentlich verbesserte VGA-Chips herstellt.	

TREND

Neue 80386-PC von Compaq und IBM	159
Innerhalb kürzester Zeit haben sowohl Compaq als auch IBM neue PC mit 80386-Prozessoren auf die Beine gestellt.	

RUBRIKEN

Editorial	3
Briefe	6
Bücher	38
Spruch des Monats	124
Zitat des Monats	63
Markt	135
Vorschau	166
Impressum	165

Abstrahlsicherheit

mc 1988, Heft 6, Seite 35

Bei der Abstrahlsicherheit (genauer: Schutz gegen die Erfassung kompromittierender Abstrahlung) handelt es sich um Maßnahmen, die die Aussendung (= Abstrahlung) modulierter elektromagnetischer Wellen sowie magnetischer Felder, die beim Betrieb von elektromechanischen bzw. elektronischen Bauteilen entstehen, verhindern bzw. auf nicht mehr erfaßbare Signalpegel reduzieren. Insofern dienen die Maßnahmen der Abwehr von Lausangriffen im elektromagnetischen Spektrum (im wesentlichen im Hochfrequenzbereich). Hauptaussender bei DV-Geräten sind der Bildschirm, sowie serielle Geräte wie Tastatur, Drucker, Modems usw. Da es sich bei den Abstrahlschutzmaßnahmen im wesentlichen um reine Hardwarelösungen handelt, ist ein Schutz gegen das „Knacken von Codes im Inhouse-Bereich“ über das Verhindern des Empfanges kompromittierender Abstrahlung hinaus nicht gegeben; Softwaremaßnahmen (wie Modifikation an Betriebssystem, Netzwerk-, DFÜ- und Datenbanksoftware) sind zwar hochaktuell, fallen aber nicht unter den Begriff der Abstrahlsicherheit.

Ebenfalls zu unterscheiden ist die Gefahr des Abhörens von öffentlichen Postnetzen, über die Daten übertragen werden: Hiergegen können Mittel der Verschlüsselung in Form von Hard- und/oder Software (Stichwort DES) wirkungsvoll eingesetzt werden.

Bei der Formulierung der Schlußaussage, daß bisher zum Erreichen der Abstrahlsicherheit „zentnerschwere Bleimäntel“ erforderlich waren, ist offensichtlich die Phantasie mit dem Autor durchgegangen; mit Radioaktivität hat diese Thematik (Gott sei Dank!) nichts zu tun.

Jeder Radio- oder Fernmelde-techniker kann Ihnen bestätigen, daß je nach Frequenzbe-

reich z. B. geerdete Metallfolien oder -gitter die Ausbreitung elektromagnetischer Wellen wirkungsvoll dämpfen bzw. unterdrücken.

Michael Krischer,
8000 München 70

Videotext in den Mikrocomputer

Nach Erscheinen des mc-Artikels und Auslieferung der ersten Bausätze sind etwa 50 Briefe bei mir angekommen. Einige Interessenten haben mich auch angerufen. Ich wurde um Informationen gebeten, die wohl im Artikel zu kurz gekommen waren. Die häufigsten Fragen möchte ich hier gleich beantworten:

Was bedeuten die Bezeichnungen I und O am V-24-Stecker?

I steht für Input und O für Output. Diese beiden Anschlüsse werden zur Zeit von der Software nicht unterstützt. Sie sind im Bedarfsfall in einer späteren Version für einen Hardware-Handshake vorgesehen. Zur weiteren Information: G steht für Ground (Masse), T für Transmit (Sendedaten) und R für Receive (Empfangsdaten).

Was hat es mit der in mc ange-deuteten „Spezial-Version“ auf sich?

Diese Version soll das Untertiteln eigener Videos ermöglichen. Es ist kein Problem, in das Bildsignal aus einer Videokamera mit Hilfe der Platine eine Schrift einzublenden.

Anders sieht es aus, wenn das zu untertitelnde Bild aus einem Videorecorder stammt: Die Synchronimpulse stehen nicht starr. Dies bewirkt, daß die eingeblendete Schrift im Bild wackelt. Abhilfe schafft die „Spezial-Version“; jedoch mit dem Nachteil, daß der Abgleich etwas kritischer wird. Da ich diese Schaltung nicht mit vielen unterschiedlichen Recordern testen konnte, habe ich auf die Veröffentlichung verzichtet.

Das Besondere an dieser Möglichkeit ist, daß Texte als normale Schrift oder mit schwarz ausgestanztem Hintergrund an jede beliebige Stelle im Bild eingeblendet werden können.

Gibt es ein Programm für einen IBM-Rechner, mit dessen Hilfe eine komfortable menuegesteuerte Bedienung der Platine möglich ist, und das auch das Ausdrucken von Videotextseiten – auch in Farbe – unterstützt?

Es wird daran gearbeitet! Vor-erst muß man sich mit einem normalen Terminalprogramm wie Procomm begnügen.

Ist das Quellprogramm auch auf Diskette erhältlich?

Nein, nur als Assemblerlisting.

Kann anstatt des SAA 5230 auch ein SAA 5231 verwendet werden?

Ja, der SAA 5231 ist der Nachfolgetyp. Der SAA 5230 wird nicht mehr gefertigt.

Wie hoch sollte der Pegel des FBAS-Signals sein?

Im Bereich 0,7...1,4 V_{eff}. Liegt der Pegel bei 1,75...3,5 V, so muß der Anschluß 2 des SAA 5231 von Masse entfernt werden.

Wo kann man den kompletten Bausatz beziehen?

Bei der Firma Elektronikladen in Detmold.

Ein Fertiggerät bietet die Firma Wiegand-Video-Daten-Systeme, Palmersdorfer Hof, 5040 Brühl, Tel.: 02232/45028 an.

Noch drei Punkte zur Ergänzung:

Mehrfachseiten können auch direkt angewählt werden. Zum Beispiel F31940005 wählt die fünfte Seite der Mehrfachseite 194 im Speicher 3 an. Der Abbruch der Befehle „A“ und „RC“ erfolgt nicht mit jeder beliebigen Taste, sondern mit CTRL C.

Besonderheit: Die Videotext Codes für Grafik-Rot (11H), Grafik-Gelb (13H) und Block

(7FH) werden in dieser Reihenfolge umgewandelt zu 0EH, 0FH und 1BH. Der Grund: Diese VT-Zeichen entsprechen XON, XOFF und DEL. Sie werden vom Terminal-Programm als Steuerzeichen erkannt und nicht als Daten aufgezeichnet. Die Einlese-Routine wandelt diese Zeichen wieder zurück.

Eckhard Schadwinkel,
5000 Köln

Quellenangaben

Ich finde den Brief von Herrn Steinort in mc 6/88 äußerst aktuell und hoffe, daß die Redaktion in Zukunft auf Quellenangaben hohen Wert legt: Zahlreiche nützliche Beiträge basieren auf in Handbüchern spezifizierten, aber nicht allgemein verwendeten Hard- und Software-„Features“.

Ein Hinweis auf das entsprechende Handbuch würde die Nützlichkeit des Beitrages erheblich erhöhen – man findet oft weitere Informationen und Hinweise, wenn man selber nachschaut.

Zu „Farbe im DOS-Alltag“ (H. Greb in mc 5/88) müßte bezüglich des PROMPT-Befehls z. B. das DOS-Referenzhandbuch und das technische DOS-Handbuch erwähnt werden. Für Benutzer von Monochrom-Bildschirmen ist es sicherlich interessant, nachzuschauen: Grafikparameter 4 = Unterstreichen. Ich verwende den PROMPT-Befehl:

```
prompt=$e[0;33;40m$t
          $h$h$h$h$h$h
          $e[0;36;40m$P$g
          $e[0;32;40m
```

um Uhrzeit, Standard-Laufwerk und Verzeichnis farbig anzuzeigen (z. B.:

19.31.C:\util\editor>). Dieser Befehl steht natürlich in AUTOEXEC.BAT, da er dann automatisch ausgeführt wird.

S. Röhlich,
7030 Böblingen



Holen Sie sich den Informationsvorsprung für den nächsten Karrieresprung.

mc ist für Leute gemacht, die ständig auf dem neuesten Wissenstand sein müssen. Denn wer Hard- und Software entwickelt, Systeme den Kundenbedürfnissen anpaßt, Firmen berät oder Computer und Programme verkauft, der braucht Informationen Marke mc. Und das regelmäßig.



Wenn Sie das unterschreiben können, zögern Sie nicht: Bestellen Sie noch heute Ihre mc mit der Karte auf dem Kartenbeihalter in dieser Ausgabe. Schließlich wollen Sie doch bald wieder Grund zum Feiern haben.

mc
Die Mikrocomputer-Zeitschrift

Ein mc-Abonnement hat Folgen: Sie werden immer besser.

Günther Sternberg

CAD für schnelle AT-Rechner

Generic-CADD im Test

Für jeden, der mit dem Gedanken spielt, ein CAD-Programm anzuschaffen, stellt sich zuerst die Frage nach der benötigten Hardware. Laut Handbuch reicht jeder IBM-PC/XT/AT-kompatibler Computer mit mindestens 384 KByte Speicher und zwei Laufwerken (360 KByte) oder einem Laufwerk und einer Festplatte zum Betrieb von Generic CADD aus. Weiterhin wird eine von Generic CADD unterstützte Grafikkarte benötigt (Tabelle 1). Das Programmpaket besteht aus einem Grundmodul und einer Reihe von Erweiterungsmodulen und Bibliotheken. Zum Test lagen uns folgende Module vor:

- Grundmodul Version 3.0
- Drafting Enhancements-1
- Drafting Enhancements-2
- AutoDimensioning
- DotPlot
- AutoConvert
- Symbolbibliothek Elektro/Elektronik
- Symbolbibliothek Elektronik I
- Symbolbibliothek Heizung/Sanitär
- Symbolbibliothek Maschinenbau

Theoretisch lassen sich alle Funktionen des Programms über die Tastatur bedienen, in der Realität kommt man aber um die Anschaffung eines Trackballs, einer Maus oder eines Digitizers nicht herum. In der Aufzählung der verfügbaren Treiber für die Eingabegeräte (Tabelle 2) kommen daher viele Digitizer vor. Mit einem Digitizer läßt sich der Bedienungskomfort des Programms erheblich steigern, da man entweder fertige oder eigene Digitizer-Menüs installieren kann. Im Grundmodul ist als Ausgabegerät der Plotter vorgesehen (Ta-

Mit der Bezeichnung CAD (Computer Aided Design) werden sehr viele Programme angeboten, deren Einsatzgebiet vom einfachen Malprogramm bis zum hochwertigen Konstruktionsprogramm reicht. Welchen Rang nimmt Generic CADD Version 3.0 in dieser Reihe ein?

Tabelle 1: Einige der aufgeführten Bildschirmkarten sind auf dem deutschen Markt nicht vertreten

AST MONOGRAPHIC PLUS 720 x 350, Monochrome	PHOTON MEGA 640 x 350, Monochrome
ATI Graphics Solution 720 x 350, Monochrome	640 x 200, 16 Color
720 x 360, Monochrome	640 x 350, 4 Color
640 x 400, Monochrome	640 x 350, 16 Color
720 x 348, Monochrome	640 x 480, 16 Color
AT&T 6300 MONO/COLOR 640 x 200, 2 Color	800 x 600, 16 Color
640 x 400, 2 Color	QuadRam ProSync EGA 640 x 350, Monochrome
EVEREX GRAPHICS EDGE 640 x 200, 2 Color	640 x 200, 16 Color
640 x 200, 4 Color	640 x 350, 4 Color
640 x 200, 16 Color	640 x 350, 16 Color
640 x 400, 4 Color	640 x 480, 16 Color
720 x 348, Monochrome	752 x 410, 16 Color
NEC GB-I MULTISYNC BOARD 640 x 350, Monochrome	SIGMA COLOR 400 640 x 200, 2 Color
640 x 200, 16 Color	640 x 400, 16 Color
640 x 350, 4 Color	STB GRAPHIX PLUS II 640 x 352, Monochrome
640 x 350, 16 Color	640 x 200, 4 Color
640 x 480, 16 Color	STB Systems Multi Res 640 x 350, Monochrome
HERCULES MONOCHROME GRAPHICS 720 x 350, Monochrome	640 x 200, 16 Color
720 x 360, Monochrome	640 x 350, 4 Color
640 x 400, Monochrome	640 x 350, 16 Color
720 x 348, Monochrome	640 x 480, 16 Color
IBM COLOR GRAPHICS ADAPTER 320 x 200, 4 Color	752 x 410, 16 Color
640 x 200, 2 Color	832 x 350, 16 Color
IBM ENHANCED GRAPHICS ADAPTER 640 x 350, Monochrome	TECMAR GRAPHICS MASTER 640 x 400, Monochrome
640 x 200, 16 Color	720 x 352, Monochrome
640 x 350, 4 Color	720 x 400, Monochrome
640 x 350, 16 Color	720 x 704, Monochrome
LEADING EDGE MODEL „D“ 720 x 352, Monochrome	640 x 200, 2 Color
640 x 200, 2 Color	640 x 200, 16 Color
640 x 200, 16 Color	640 x 400, 16 Color
320 x 200, 4 Color	720 x 400, 4 Color
PERSYST BoB/16 640 x 200, 2 Color	800 x 480, 4 Color
640 x 400, 16 Color	VEGA 7 Deluxe EGA 640 x 350, Monochrome
PERSYST BoB/MG 640 x 200, 2 Color	640 x 200, 16 Color
	640 x 350, 4 Color
	640 x 350, 16 Color
	640 x 480, 16 Color
	752 x 410, 16 Color
	WYSE WY-700 HIGH RES 1280 x 800, Monochrome

belle 3). Mit dem Zusatzmodul DotPlot läßt sich eine Zeichnung auch auf Nadeldruckern (Tabelle 4) ausgeben.

Die Installation des Grundmoduls erfolgt durch ein komfortables menügesteuertes Programm. Die zugehörigen Erklärungen im deutschen Handbuch sind recht ordentlich, in manchen Punkten für einen Anfänger aber nicht ausreichend genug.

Ein leistungsfähiger Rechner ist notwendig

Zu der vorher erwähnten „Minimalkonfiguration“ muß ich jedoch einschränkend einige Bemerkungen anbringen. Fast alle CAD-Programme arbeiten nach dem sogenannten Elementverfahren. Dies bedeutet, daß die Elemente einer Zeichnung (Linie, Kreis etc.) nicht in Form eines Bitmusters (Pixelgrafik) gespeichert werden, sondern nur die für jedes Element spezifischen Parameter (z. B. Anfangs- und Endekoordinate einer Linie). Soll ein CAD-Programm möglichst universell nutzbar sein, so zum Beispiel zur Anfertigung einer maßstabgetreuen Zeichnung eines Autos, so müssen die meisten Parameter in Form von Real-Zahlen gespeichert werden, da mit Integer-Zahlen die Forderung nach großer Zeichenfläche und hoher Auflösung nicht unter einen Hut zu bringen ist. Diese rechenzeitintensive Real-Arithmetik macht sich besonders beim Bildaufbau bemerkbar. Die Ausgabe der Komplettansicht eines Schaltplans für eine arithmetische Logikeinheit (ähnlich dem TTL-Baustein 74181) in Gatterform, bestehend aus 1362 Linien und 39 Textzeichen, benötigte auf meinem Testrechner (6 MHz-AT

TEST

Tabelle 2: Die Palette der Eingabegeräte reicht vom einfachen und billigen Trackball bis zum teuren Digitizer

CALCOMP 2000 Digitizer	LYNX SERIAL TRACKBALL
CALCOMP 2100 Digitizer	MAYNARD SERIAL MOUSE
CALCOMP 2500 Digitizer	MAYNARD BUS MOUSE
CALCOMP 9100 Digitizer	MICROGRID DIGITIZER
FULCRUM TRACKBALL	MICROSOFT BUS
GP-7/MARK2 SONIC Digitizer	MOUSE (VER 6.0)
GP-8 SONIC Digitizer	MICROSOFT SERIAL MOUSE
GP-8 SONIC (FLIPPED)	NUMONICS 2200 DIGITIZER
MITSUBISHI GRAFNET MODEL-01	MOUSE SYSTEMS PC BUS MOUSE
GTCO TYPE 5 Digitizer	MOUSE SYSTEMS PC MOUSE
GTCO Micro DIGI-PAD (7)	KURTA PENMOUSE +
HIPAD DT11 Digitizer	SUMMAGRAPHIC MM 1201
HIPAD DT11/AA Digitizer	SUMMAGRAPHIC MM 1812
HITACHI TIGER Digitizer	SUMMA 1812/UIOF FORMAT
IMSI MOUSE	SUMMAGRAPHIC BIT PAD 2
KURTA SERIES ONE Digitizer	SUMMA BIT PAD 2 (UIOF)
KURTA SERIES TWO Digitizer	TALOS 800 Series DIGITIZER
LOGITECH LOGIMOUSE C7	TORINGTON MANAGER MOUSE
LOGITECH BUS MOUSE	H.I. TRUE GRID Digitizer

Tabelle 3: Mit einem HP-GL-kompatiblen Plotter liegt man fast immer richtig

ALPHA MERICS	H.P. 7475A PLOTTER
ALPHAPLOT II	H.P. 7550A PLOTTER
AMDEK AMPLOT-II PLOTTER	H.P. 7570A PLOTTER
APPLE 410 PLOTTER	H.P. 7580B PLOTTER
COMSCRIBER 1 PLOTTER	H.P. 7585B PLOTTER
EPSON HI-80 PLOTTER	IOLINE LP3700 PLOTTER
GRAFTEC X-Y PLOTTER	IOLINE LP4000 PLOTTER
H.I. DMP-695 PLOTTER	NUMONICS 5424 PLOTTER
H.I. DMP-29 PLOTTER	NUMONICS 5460 PLOTTER
H.I. DMP-40 PLOTTER	NUMONICS 5860 PLOTTER
H.I. DMP-41 PLOTTER	PANASONIC DIGITAL PLOTTER
H.I. DMP-42 PLOTTER	ROLAND DXY-101 PLOTTER
H.I. DMP-51 PLOTTER	ROLAND DXY-800 PLOTTER
H.I. DMP-52 PLOTTER	SWEET-P 100 PLOTTER
H.I. DMP-595 PLOTTER	SWEET-P SIX-SHOOTER
H.P. 7220 PLOTTER	TALLY PIXY-3 PLOTTER
H.P. 7440A PLOTTER	TANDY FP-215 PLOTTER
H.P. 7470A PLOTTER	

Tabelle 4: Fast alle namhaften Druckerhersteller sind in der Liste aufgeführt

AMT OFFICE PRINTER - HIGH COLOR	FACIT 4528	MPI - MED. RES.
AMT OFFICE PRINTER - HIGH RES.	FACIT 4542	MTS80
AMT OFFICE PRINTER - LOW RES.	FUJITSU 24C - HIGH RES.	NEC 8023
AMT OFFICE PRINTER - MED. RES.	FUJITSU 24C - HIGH RES. COLOR	NEC 8027A
AMT OFFICE PRINTER - V HIGH RES	FUJITSU 24C - V. HIGH RES.	NEC P2/P3 - COLOR HIGH RES
AMT OFFICE PRINTER - V LOW RES	FUJITSU 24C - V. HIGH RES. COLOR	NEC P2/P3 - COLOR LOW RES.
AMT OFFICE PRINTER - V V HIGH	FUJITSU 24D - HIGH RES.	NEC P2/P3 - COLOR MED RES.
AMT OFFICE PRINTER - VH COLOR	FUJITSU 24D - LOW RES.	NEC P2/P3 - COLOR V. HIGH
AMT OFFICE PRINTER - VL COLOR	FUJITSU 24D - MES RES.	NEC P2/P3 - HIGH RES.
AMT OFFICE PRINTER - VVH COLOR	GENICOM 3180-3404 SERIES	NEC P2/P3 - LOW RES.
AMT OFFICE PRT - COLOR LOW	GORILLA BANANA	NEC P2/P3 - MED. RES.
AMT OFFICE PRT - COLOR MED.	HEWLETT PACKARD	NEC P2/P3 - V. HIGH RES.
ANADIX - HIGH RES.	HP LASERJET - 100 DPI	NEC P5 - VERY HIGH RES.
ANADIX - HIGH RES. COLOR	HP LASERJET - 150 DPI	NEC P5 COLOR - VERY HIGH
ANADIX - LOW RES.	HP LASERJET - 300 DPI	NORTH ATLANTIC QUANTEX - HIGH
ANADIX - LOW RES. COLOR	HP LASERJET - 75 DPI	NORTH ATLANTIC QUANTEX - LOW
BROTHER 2024L - 24 PIN	IBM COLOR PRINTER - LOW RES.	NORTH ATLANTIC QUANTEX - MED
BROTHER TWINRITER - HIGH	IBM COLOR PRINTER - MED. RES.	OKIDATA 2410
BROTHER TWINRITER - LOW	IBM GRAPHICS PRINTER - HIGH	OKIMATE 20 - B&W
BROTHER TWINRITER - MED.	IBM GRAPHICS PRINTER - LOW RES.	PANASONIC - HIGH RES.
BROTHER TWINRITER - V. HIGH	IBM GRAPHICS PRINTER - MED.	PANASONIC - LOW RES.
CANON PJ1080A COLOR INKJET	IBM GRAPHICS PRINTER - VERY HIGH	PANASONIC - MED RES.
CENTRONICS	IBM INKJET - BLACK & WHITE	PANASONIC - VERY HIGH RES.
CITOH - HIGH RES.	IBM INKJET - COLOR	QUADJET BLACK & WHITE
CITOH - LOW RES.	IDS - B&W	QUADJET COLOR
CITOH - MED. RES.	IDS 440	RADIO SHACK - MOST RS PRINTERS
CITOH - VERY HIGH RES.	IDS PRISM - COLOR	RADIO SHACK 2100 - 24 PIN
DATAPRODUCTS 8050	INTEGREX COLOUR JET 132	RADIO SHACK 430 - HIGH
DATASOUTH - HIGH RES.	JDL 750 - HIGH RESOLUTION	RADIO SHACK 430 - LOW
DATASOUTH - LOW RES.	JDL 750 - HIGH RESOLUTION COLOR	RADIO SHACK 430 - MEDIUM
DEC LA50, LA100 - HIGH RES.	JDL 750 - LOW RESOLUTION	RADIO SHACK CGP-220 B&W
DEC LA50, LA100 LOW RESOLUTION	JDL 750 - LOW RESOLUTION COLOR	RADIO SHK CGP-220 COLOR
DIABLO S32	JDL 750 - MED. RESOLUTION COLOR	SEIKOSHA
EPSON - HIGH RES.	JDL 750 - MEDIUM RESOLUTION	STAR MICRONICS - HIGH RES.
EPSON - LOW RES.	MALIBU	STAR MICRONICS - LOW RES.
EPSON - MED. RES.	MANN. TALLY SPIRIT 80 - HIGH	STAR MICRONICS - MED. RES.
EPSON - VERY HIGH RES.	MANN. TALLY SPIRIT 80 - MEDIUM	STAR MICRONICS - VERY HIGH RES.
EPSON GQ-3500 LASER PRINTER	MANNESMAN TALLY 160 - HIGH RES.	STAR SB-10 - HIGH RES.
EPSON JX80 - HIGH RES.	MANNESMAN TALLY 160 - LOW RES.	TOSHIBA - HIGH RES.
EPSON JX80 - LOW RES. COLOR	MANNESMAN TALLY 160 - MED. RES.	TOSHIBA - HIGH RES. COLOR
EPSON JX80 - MED. RES.	MANNESMAN TALLY 420	TOSHIBA - V. HIGH RES COLOR
EPSON JX80 - VERY HIGH RES.	MANNESMAN TALLY SPIRIT 80 - LOW	TOSHIBA - VERY HIGH RES
EPSON LQ1500 - 24 PIN	MPI - HIGH RES.	TOSHIBA 1350 - HIGH RES.
EPSON LQ2500 - COLOR	MPI - LOW RES.	

Lesen Sie weiter auf Seite 44.

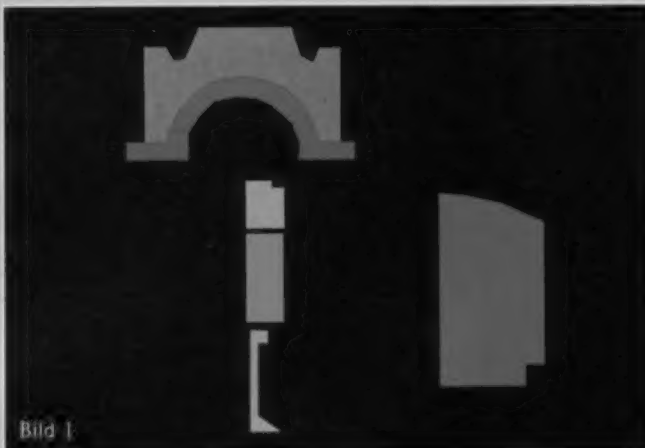


Bild 1

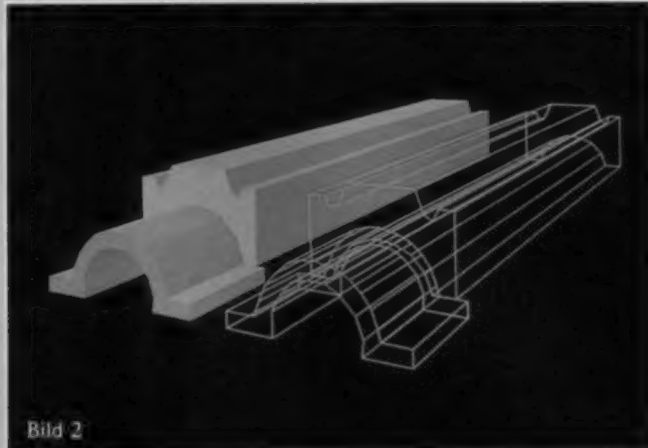


Bild 2

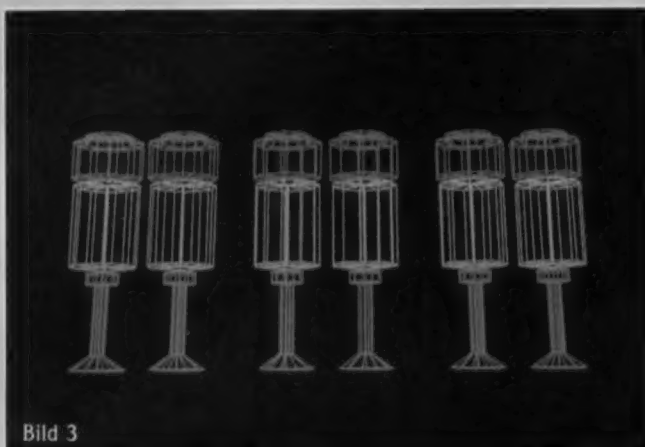


Bild 3

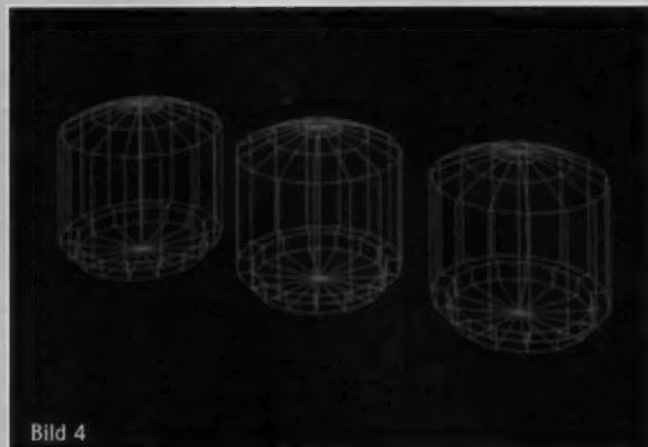


Bild 4

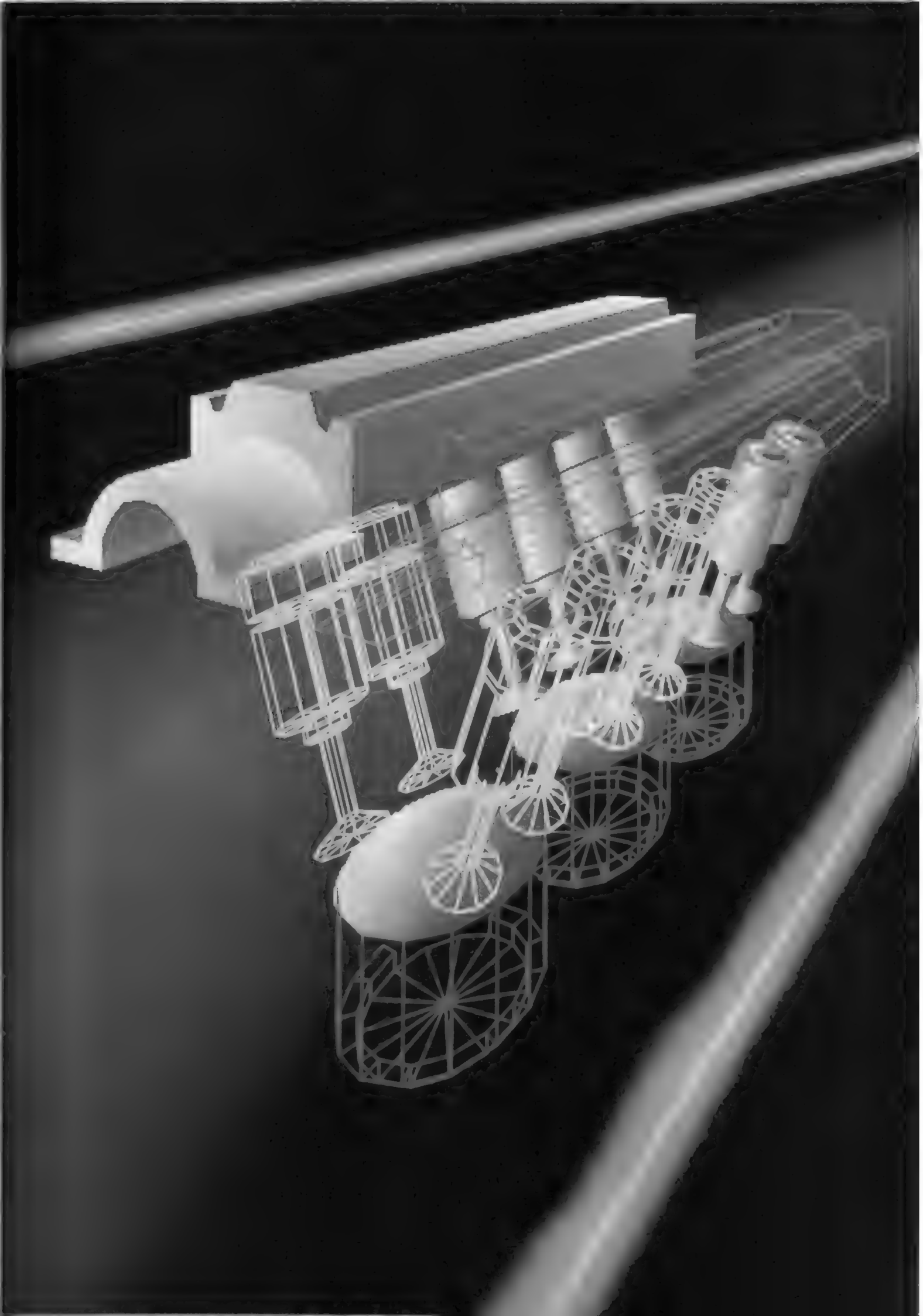
Das Programm, mit dem mein Studio arbeitet, ist ein Computergrafiksystem mit CAD-orientierten Konstruktionshilfen – basierend auf einem AT-Rechner. Die Art-Direktion des Franzis-Verlages beauftragte mich, darauf ein Titelbild zu entwerfen, das für den Begriff CAD symbolträchtig sei. Als Bildmotiv waren Detailspekte von Automobilen (und anderen High-Tech-Gegenständen) im Gespräch. Ich entschied mich, „einen Motor zu bauen“. Im ersten Entwurfs-Schritt habe ich Grundelemente mit Hilfe von Polygonen (Vielecken) in der Ebene konstruiert (Bild 1). Beim nächsten Schritt wurden die Grundelemente mit zwei unterschiedlichen Methoden in die dritte Dimension gebracht. Zuerst wurden die Deckelflächen durch Ausdehnung „in die Tiefe“ erzeugt. Dabei wurde jeder Punkt der flächigen Ausgangsfigur um einen passenden Wert in Z-Richtung „hinter den Bildschirm“ verschoben und dort erneut gesetzt (X,Y und Z-Koordinate können bis zum Wert 36 000 genutzt werden.) Dies ergab einen räumlichen Körper, der zuerst als Gittermodell dargestellt wurde. Von diesem Körper wurde dann mit dem Programm ein Duplikat angefertigt und im Raum links neben den Originalkörper gestellt. Dieses Duplikat erhielt im nächsten Arbeitsschritt anstelle der Gitterstruktur eine gefüllte Oberfläche, wobei schon Licht und Schattenwirkung berücksichtigt wurden (Bild 2).

Ein Titelbild entsteht

Alle übrigen Grundfiguren erhielten ihre Dreidimensionalität durch die zweite Methode, durch Rotation, wobei eine jede Grundfigur um die vorgesehene Achse in 18 Schritten rotiert wurde. Daraus ergaben sich die Ventilelemente (Bild 3). Ein komplettes Element wurde anschließend noch fünfmal dupliziert, um zu einer Sechser-Reihe zu kommen. Diese Reihe wurde ebenfalls dupliziert. Da diese beiden Ventilreihen nun im rechten Winkel zu den oberen Deckelfiguren standen, mußten sie um 90 Grad (um die Y-Achse) gedreht werden, wenn sie parallel zur Deckelfigur stehen sollten. Um die beiden Reihen nun noch etwas schräg zueinander zu stellen, mußten sie durch eine 20-Grad-Drehung um die Z-Achse je in positiver und negativer Drehrichtung in Position gebracht werden. Im letzten Konstruktionsschritt entstanden die Kolben, ebenfalls durch Rotation. Der komplette Kolben wurde anschließend noch zweimal dupliziert. Diese drei Kolben wurden dann in linearer Anordnung nebeneinander gestellt und durch eine 90-Grad-Drehung um die Y-Achse parallel zu den übrigen Elementen

ten gesetzt. Zu diesem Zeitpunkt waren alle Elemente konstruiert und in Frontalansicht zu betrachten. Zur Überprüfung der Richtigkeit von Position und Proportion der Elemente wurde das Gesamtbild jeweils um 90 Grad um die einzelnen Achsen gedreht (Ansicht von Oben/von der Seite). Um nun zum endgültigen Bild zu kommen, wurde der Gesamtkörper wieder in die Frontalansicht gebracht und komplett um 20 Grad in der Y-Achse in negativer Richtung gedreht. Die Elemente, die nicht als Gittermodell erscheinen sollten, wurden danach als Einzelgruppen flächig schattiert. In einem darauffolgenden Arbeitsgang wurden diese Teile in einem speziellen Illustrationsprogramm mit einer weichen Licht-/Schatten-Wirkung ausgestattet. Alle übrigen Teile wurden in dem Gitterzustand belassen. Zuletzt wurde noch der Abstand vom Betrachter zum Objekt gewählt (diese Funktion beeinflusst den Grad der perspektivischen Darstellung; je weiter der Abstand zum Objekt, desto paralleler wird die Perspektive). In diesem Fall ein relativ geringer Abstand, um eine starke Fluchtperspektive zu erhalten. Der Fluchtpunkt selbst wurde in die rechte obere Bildecke gelegt. Zum Schluß mußte der komplette Motor nur noch in den bereits mit einem anderen Programmtitel (Brush-Programm) erstellten Hintergrund mit Farbverläufen einkopiert werden.

Erich Jäger



mit EGA-Karte und numerischen Coprozessor) ungefähr 20 Sekunden. Dabei sind in dieser Zeichnung aber noch keine der besonders zeitintensiven Funktionen, wie Splines oder Schraffuren, enthalten.

Generic CADD speichert alle Daten zu einer in Bearbeitung befindlichen Zeichnung im Hauptspeicher ab. Je größer also der Hauptspeicher ist, desto komplexer kann die Zeichnung sein. Beim Betrieb von Generic CADD mit den oben genannten Zusatzmodulen blieben auf einem AT mit 640 KByte Hauptspeicher 230 KByte für die Zeichnungsdaten übrig. Das vorher erwähnte Beispiel belegt knapp 50 KByte Speicher. Speicher oberhalb der 640 KByte-Grenze (z. B. EMS-Speicherkarten) werden nicht unterstützt.

Wer also intensiver mit Generic CADD arbeiten möchte und nicht über Zeit im Überfluß verfügt, der sollte als Minimalkonfiguration einen schnellen AT mit 640 KByte Hauptspeicher, Festplatte und numerischen Coprozessor ansetzen.

Flexible Befehlsauswahl

Eine ideale Methode zur Befehlsauswahl schlechthin gibt es nicht und kann es nicht geben. Eine Gruppe der Anwender wählt einen Befehl am liebsten über die Tastatur aus, während eine andere Gruppe möglichst alle Operationen mit Hilfe der Maus oder des Digitizers erledigt. Generic CADD stellt beide Anwendergruppen zufrieden. Jeder Befehl kann entweder durch die Eingabe eines zweistelligen Namens Kürzels (*Bild 1*) über die Tastatur, durch Anklicken des entsprechenden Punkts eines Digitizer-Menüs oder durch das Menü am rechten Rand des Bildschirms aktiviert werden (*Bild 2*). Das Bildschirmmenü unterteilt sich in ein Hauptmenü und eine Reihe von Untermenüs. Es läßt sich zugunsten einer größeren Zeichenfläche wahlweise ausblenden. Alle Möglichkeiten der Befehls Eingabe lassen sich beliebig mischen.

Damit sind die Möglichkeiten der Benutzersteuerung aber noch nicht ausgeschöpft. Der Anwender kann neben den Digitizer-Menüs auch die Bildschirmen nach seinen eigenen Wünschen konfigurieren. Somit kann jeder Anwender die von ihm am häufigsten benötigten Befehle in eigenen Menüs zusammenfassen. Jedem Menüpunkt und jeder Funktionstaste der Tastatur können auch ganze Befehlssequenzen zugeordnet werden. Die Konfiguration der Menüs und Tasten geschieht auf einfache Weise und ist im Handbuch detailliert beschrieben.

Eigenschaften des Grundmoduls

Das Grundmodul von Generic CADD Ver-

sion 3.0 kennt als Zeichnungselemente Linien, Rechtecke, gleichseitige Vielecke, Kreise und Kreisbögen, Ellipsen, Splines, Symbole und Texte. Für die Linien lassen sich verschiedene Farben, Typen (strichliert etc.) und Strichbreiten einstellen. Beim Ziehen einer Linie oder eines Linienzugs kann man wahlweise die Ausgabe einer „Gummilinie“ ein- oder ausschalten. Zur Beschriftung der Zeichnung stehen sieben verschiedenen Zeichensätzen zur Verfügung. Jede Zeichnung kann aus maximal 256 Ebenen bestehen. Die Ebenen lassen sich einzeln oder in beliebiger Kombination anzeigen. Jede Ebene kann gelöscht, geändert, gedreht, neu skaliert, geladen oder gespeichert werden. Als Maßeinheit für Koordinatenangaben, Rasterabstand, Textgröße etc. kann man zwischen metrischen Angaben und Zoll-Angaben wählen. Eine interessante Möglichkeit eröffnet sich durch die manuelle Koordinateneingabe. Bei einigen Anwendungen ist es oft leichter eine Kurve durch manuelle Eingabe der Eckpunkte zu beschreiben als mit Hilfe der Maus. Selbsterstellte Symbole oder Symbole aus einer gekauften Bibliothek können vor der Platzierung gedreht und vergrößert bzw. verkleinert werden. Man kann ein Symbol entweder als eine Einheit in die Zeichnung übernehmen oder es bei der Übernahme in seine Einzelelemente auflösen. Im ersten Fall erhält jedes Symbol einen sogenannten Anfaßpunkt, mit dessen Hilfe sich das Symbol auch nachträglich wieder selektieren läßt. Auf diese Weise kann ein Symbol in seiner ganzen Einheit zum Beispiel verschoben oder gelöscht werden.

Die Zusatzmodule

Das Zusatzmodul DotPlot zur Ausgabe einer Zeichnung auf einem Nadeldrucker wurde bereits erwähnt. Mit ihm läßt sich die ganze Zeichnung oder ein beliebiger Ausschnitt der Zeichnung ausgeben. Die Auswahl eines Ausschnitts kann grafisch erfolgen.

Das Zusatzmodul AutoConvert soll den Austausch von Zeichnungen mit anderen CAD-Programmen ermöglichen. Es konvertiert eine Zeichnung von Generic CADD in das DXF-Format, ein Datenformat zur Beschreibung von Zeichnungen, daß durch AutoCAD eingeführt wurde. Umgekehrt kann eine Datei im DXF-Format zu einer Generic CADD Datei gewandelt werden.

Die Erweiterung AutoDimensioning enthält Funktionen zur automatischen Bemaßung einer Zeichnung. Es können Längs-, Winkel- und Radiusmaße erstellt werden. Es lassen sich Maß- und Maßhilfslinien zeich-

nen und mit den entsprechenden Maßzahlen versehen. Es ist eine Einzel-, Ketten- oder Bezugsbemaßung möglich. Die Maßzahl läßt sich parallel zur Maßlinie bzw. oberhalb oder unterhalb der Maßlinie anbringen. Mit dieser Aufzählung sind längst nicht alle Möglichkeiten dieses Moduls hinreichend beschrieben.

Das Modul Drafting Enhancements-1 enthält eine Reihe nützlicher Funktionen zur Erweiterung der Snap- und Editierbefehle. So läßt sich beispielsweise automatisch der Schnittpunkt zweier Elemente berechnen, die Lotrechte zu einem Element bestimmen, parallele Linien zu einer vorgegebenen Linie zeichnen, der Mittelpunkt von Linien oder Bögen berechnen und einiges mehr.

Drafting Enhancements-2 enthält Funktionen zum Schraffieren und Flächenfüllen. Neben einer Reihe standardmäßiger Schraffur- und Füllmuster lassen sich auch selbstdefinierte Muster verwenden.

Die Symbolbibliotheken

Zum Test lagen uns Symbolbibliotheken aus den Bereichen Heizung/Sanitär, Maschinenbau und Elektro/Elektronik vor. Die Bibliothek Elektro/Elektronik enthält 135 Symbole, angefangen vom Widerstand über die DIN-Schraube bis zum Schaltzeichen eines Mikrowellenherds. Alle Symbole sind in der zugehörigen Dokumentation namentlich und grafisch aufgeführt.

Ein kleiner Nachteil der Symbolbibliotheken sei nicht verschwiegen. Jedes Symbol wird in einer eigenen Datei gespeichert. Durch die Installation der Bibliotheken aus dem Bereich Elektrotechnik füllt sich die Festplatte gleich um einige Hundert Dateien. Das ellenlange Inhaltsverzeichnis bei der Suche nach einem bestimmten Symbol wäre ja noch zu ertragen, würde nicht jedes auch noch so kleine Symbol immer mindestens einen Cluster (dies sind auf einer Festplatte 2048 Bytes) belegen. Bei Symbolen mit wenigen Elementen (z. B. ein Widerstand) wird auf diese Weise oft 90% Speicherkapazität verschenkt.

Uneinheitliche Qualität der Dokumentation

Das deutsche Handbuch zum Grundmodul ist im Großen und Ganzen mit Gut zu bewerten. Das Handbuch besteht aus einer Loseblatt-Sammlung in einem Ringbuch. Am Anfang des Handbuch befindet sich eine knappe Einführung zum Thema CAD und ein kleines Tutorium. Jeder Befehl des Grundmoduls ist auf ein oder zwei Seiten relativ ausführlich und übersichtlich be-

Zeichnen		Zeichnung	
(PO)	Standard-Punkt	(DL)	Zeichnung laden
(LI)	Gerade Linie	(DS)	Zeichnung sichern
(RE)	Rechteck	(DP)	Zeichnung plotten
(RP)	gleichseit. Vieleck	(DA)	Zeichnung ausrichten
(C2)	Zweipunkt-Kreis	(DX)	Zeichnung löschen
(C3)	Dreipunkt-Kreis	(DG)	Attribute ändern
(A3)	Dreipunkt-Bogen	(DR)	Zeichnung drehen
(A4)	Vierpunkt-Bogen	(DZ)	Neuer Maßstab
(EP)	Ellipse	(DO)	neuer Ursprung
(CV)	Kompl. Kurve/Spline		
Symbole		Steuerungsbefehle	
(CC)	Symbol definieren	(TO)	Fangradius
(CZ)	Symbol Maßstab	(NP)	Fange nächsten Punkt
(CR)	Symbol drehen	(AL)	Alle Ebenen
(CP)	Symbol platzieren	(OR)	Rechtwinklige Linien
(CE)	Symbol auflösen	(LS)	Zeichnungs-Grenzen
(CI)	Symbol abbilden	(BP)	Basispunkt
(CX)	Symbol löschen	(TM)	Abtastmodus
(CS)	Symbol sichern	(SC)	Fange benachbarten Punkt
(CL)	Symbol laden/	(RZ)	Skalierung im Abtastmodus
	Symbole auflisten	(GC)	Einrasten auf Komponentenpunkte
(CN)	Symbole austauschen		
Text		Raster	
(PS)	Zeichensatz auswählen	(GS)	Raster-Größe
(TK)	Text-Farbe	(SG)	Fange Rasterpunkt
(TS)	Text-Größe	(GR)	Raster an/aus
(TR)	Text-Drehung		
(TA)	Text-Propotion	Anzeige	
(TS)	Text-Neigung	(VM)	Bildschirmmenü anzeigen
(TP)	Text-Platzierung	(PR)	Bezugspunkte anzeigen
(TI)	Text einfügen	(PC)	Konstruktionspunkte anzeigen
(TD)	Text löschen	(PS)	Standardpunkt anzeigen
(TX)	Text ersetzen	(AC)	Absolute Koordinaten
(TC)	Zeichen erstellen	(DC)	Relative Koordinaten
		(SL)	Statuszeile
Darstellungsgröße		(RB)	Gummiband
(ZM)	Zoomen	(DK)	Farbe Bildschirmtexte
(ZA)	alles anzeigen	(SR)	Schirm Proportionen
(ZL)	Zoomen Zeichnungsgrenzen	(CK)	Cursorfarbe
(ZW)	Ausschnitt zoomen	(CU)	Cursorgröße
(ZV)	gespeicherte Ansichten laden	(SP)	Wechsel Grafik/Text
(ZP)	vorheriger Zoomfaktor	(LZ)	Skalierung der Linienteile
(ZU)	Zweifach Zoom	(FA)	Schnelle Kreise
(ZB)	Zoom zurück	(TF)	Schneller Text
(PA)	Bildausschnitt bewegen	(TV)	Text anzeigen
(RD)	neu zeichnen		
(DR)	Rückwärts neuzeichnen	Einheiten	
(NV)	Ansichten abspeichern	(ME)	metrisches Maß
Editieren		(MT)	Meter
(OB)	Elemententeile löschen	(MC)	Centimeter
(OG)	Element ändern	(MM)	Millimeter
(OE)	Element löschen	(FI)	Fuß und Zoll
(OM)	Element verschieben	(PT)	Bruch-Format
(OC)	Element kopieren	(PT)	Polar-Koord.-Format
(MP)	Punkt verschieben	(MI)	Bögen und Winkel in Minuten
(EL)	Letztes löschen	(PV)	Genauigkeit Brüche
(LK)	Linien-Farbe	(DV)	Nachkomma-Stellen
(LT)	Linien-Typ	(FD)	Dezimal-Format
(LW)	Linien-Breite		
Fenster		Utilities	
(WG)	Attribute im Fenster ändern	(UE)	Zurückholen eines gelöschten Elements
(WE)	Fenster löschen	(QU)	Programm beenden
(WM)	Fenster verschieben	(MO)	Manuelle Eingabe bezogen auf
(WC)	Fenster kopieren		Zeichnungsursprung
(WI)	Fenster spiegeln	(MB)	Manuelle Eingabe bezogen auf den
(WR)	Fenster drehen		Bezugspunkt
(WE)	Fenster neuer Maßstab	(MR)	Manuelle Eingabe bezogen auf den
(WS)	Fenster sichern		letzten Punkt
(WT)	Fenster Text	(PM)	Digitizer-Fläche
		(SD)	Dig.-Menü auswählen
Ebenen		(LD)	Dig.-Menü laden
(YC)	aktuelle Ebene	(LV)	Bildschirmmenü laden
(YD)	Ebenen anzeigen	(SB)	Batch-Datei sichern
(YH)	Ebene ausblenden	(LB)	Batch-Datei laden
(YK)	Ebene löschen	(CD)	Symbole abspeichern
(YG)	Ebene ändern	(MX)	Menü entfernen
(YR)	Ebene drehen	(PD)	Speicher löschen
(YZ)	Ebene neuer Maßstab		
(YL)	Ebene laden	Messen	
(YS)	Ebene sichern	(MD)	Strecke messen
		(MA)	Winkel messen
		(MV)	Fläche messen

Bild 1. Zu diesen Befehlen des Grundmoduls kommen die Funktionen der Erweiterungsmodule noch hinzu

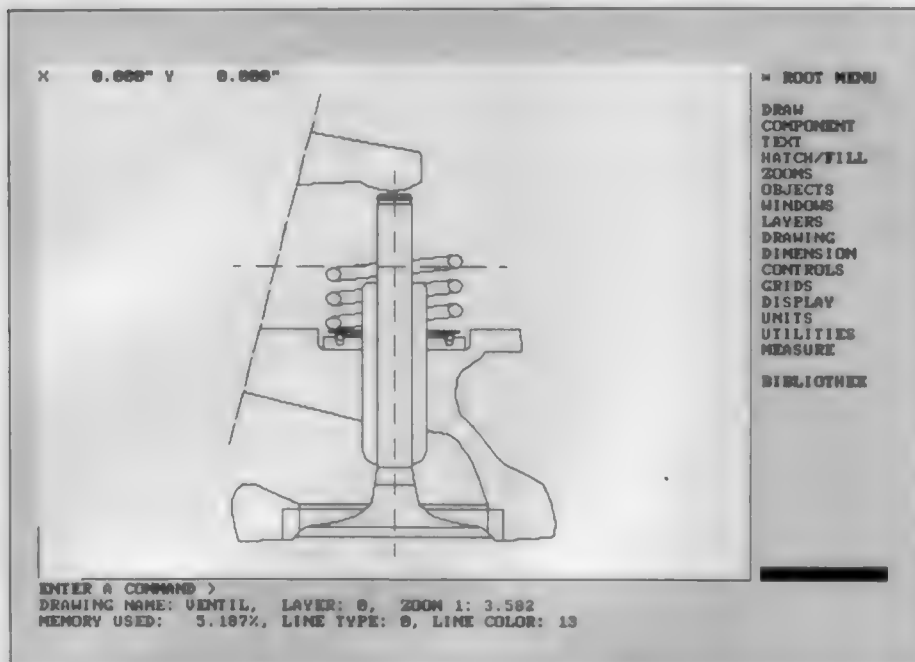


Bild 2. Einfache Benutzerführung ist für Generic CADD kein Fremdwort

schrieben. Im Anhang werden alle Installationen und Konfigurationen beschrieben. Jedes Zusatzmodul und jede Symbolbibliothek enthält sein eigenes Handbuch, wobei die Blätter der Zusatzmodule im Ordner des Grundmoduls abzulegen sind. Die Dokumentation der Zusatzmodule ist qualitativ schlechter. Die Beschreibung der doch sehr

speziellen Funktionen einiger Zusatzmodule fällt oft sehr knapp aus.

Ein leistungsfähiges Programmpaket

Die Vielzahl der Programmfunktionen und die Möglichkeit, das Programm weitgehend

den individuellen Wünschen in Bezug auf die Benutzeroberfläche anpassen zu können, sind positiv zu beurteilen. Auf Grund der Komplexität des Programmpakets sind für eine vollständige Einarbeitung doch einige Tage zu veranschlagen. Generic CADD eignet sich besonders zur Anfertigung zweidimensionaler Zeichnungen (Baupläne, Maßzeichnungen etc.). Für dreidimensionale Gestaltung oder für perspektivische Darstellungen ist Generic CADD nicht geeignet. Die niedrige Geschwindigkeit beim Bildaufbau ist bei einigen Anwendungen (zum Beispiel beim Zeichnen von Schaltplänen) störend. Auch enthält das Programm keine speziellen Funktionen für die Elektrotechnik, wie zum Beispiel die Möglichkeit Stück- und Verbindungslisten erstellen zu können. Für mich ist es jedoch völlig unverständlich, daß in Generic CADD ein Int24-Handler fehlt. So führt zum Beispiel die Angabe eines nichtbereiteten Laufwerks zu der Fehlermeldung „Abort, Retry, Ignore“, die im Grafikmodus auf dem Bildschirm in Form wirrer Striche erscheint. Ein ungeübter Anwender tut sich in diesem Falle schwer, richtig zu reagieren.

Alles in allem ist das Programmpaket aber als gelungen und empfehlenswert einzustufen. Es wird in Deutschland von PublicSoft, einem Unternehmen der Bertelsmann-Gruppe, vertrieben.

Das „SC“ in der Produktbezeichnung des Pascal-Compilers für den Atari ST des Teubner-Verlages steht für „Scientific Computation“, also wissenschaftliches Rechnen. Damit ist auch schon die Anwender-Gruppe beschrieben. Wer komplexe mathematische Probleme in Pascal lösen möchte, sollte sich Pascal-SC ansehen. Die Mathematik-Routinen rechnen nicht einfach mit 13 Digits genau, sondern wenden optimale Verfahren an. Zusätzlich kann die Rundungsrichtung vorgegeben und der Rundungsfehler festgestellt werden. In der Library stehen mächtige Operatoren zur Verfügung. Eigene Operatoren zu definieren, ist wie in Algol 68 (oder ADA) möglich. Das geschieht in einer funktionsähnlichen Form. Ein beispielsweise so definierter Operator „hoch“ erlaubt dann Schreibweisen wie „3 hoch 7“ anstatt der auf einer funktion basierenden Notation „hoch(3,7)“. Gleiche Flexibilität ist bei den Datentypen gegeben. Sind zum Beispiel A eine Matrix und b, c und x Vektoren, so ergibt die Schreibweise „c:=A*x+b“ tatsächlich das Ergebnis, das ein Mathematiker in diesem Fall erwartet. Daß darüber

Pascal-SC: Mathematik auf dem Atari ST

hinaus zahlreiche Funktionen zur Lösung komplexer mathematischer Probleme in den Libraries zur Verfügung stehen, versteht sich von selbst. Das Pascal selbst ist Ur-Pascal, wie es in modernen Dialekten schon längst nicht mehr praktiziert wird. So fehlt denn auch der Typ String, was einen zwingt, so selbstverständliche Dinge wie „readln(s)“ selbst zu programmieren. Tatsächlich gelang es mir nicht, auch nur ein einziges Programm in meiner wirklich nicht kleinen Sammlung zu finden, daß ohne drastische Änderungen unter Pascal-SC lief. Zu dem Paket gehört ein Editor mit eingebautem Syntax-Prüfer. Dieses unter TOS laufende Programm ist gewöhnungsbedürftig, da man erst diverse Tastatur-Kom-

mandos lernen muß. Der Syntax-Prüfer ist manchmal lästig, aber auch abschaltbar. Für die Einzelprogramme steht eine recht komfortable GEM-Shell zur Verfügung, diese ist aber noch nicht ganz stabil. Effekte wie „TOS-Schirm mit GEM-Menüs“ oder Probleme mit Tempus wurden im Test beobachtet. Der Zugriff auf alle System-Routinen ist möglich, da passende Library- und Include-Dateien mitgeliefert werden. Diese sind jedoch nicht im Handbuch dokumentiert, so daß sich der Anwender anhand der Include-Dateien orientieren muß. Zum Handbuch selbst: Wer als Student Bücher aus dem Verlag Teubner gelesen hat, wird feststellen, daß sich inzwischen nur eines geändert hat: Der Text über 180 Seiten ist englisch. Der trockene, mühsam zu lesende aber stets wissenschaftlich korrekte Stil ist geblieben. Bleibt als Fazit: Pascal-SC kann Mathematikern empfohlen werden, denen es auf die Äußerlichkeiten eines Programms nicht ankommt. In Standard-Anwendungen ist ST-Pascal klar überlegen und auch in den Turn Around Zeiten rund doppelt so schnell.

Peter Wollschlaeger

Peter Wollschlaeger

GFA-Basic 3.0 für Atari ST

Beim GFA-Basic muß man neuerdings noch Atari ST dazusagen, denn die Amiga-Version dieses Basic ist schon fast fertig. Da wäre dann der direkte Konkurrent zu Microsoft. So gesehen ist es wohl erlaubt, auch die neue Version für den Atari ST stellenweise mit dem Industriestandard zu vergleichen, zumal sich der Atari ST ja inzwischen auch im kommerziellen Bereich kräftig etabliert hat.

Was ist neu?

Die Frage „Was ist neu?“ kann ich als Tester nur mit „zuviel“ beantworten. Tatsächlich wird man vom Funktions-Umfang dieses Basic-Dialektes nahezu erschlagen. Das fällt natürlich besonders auf, wenn man die Aufgabe hat, das Thema so aufzubereiten, daß daraus ein Artikel in mc-Qualität entsteht und nichts Wesentliches fehlt. Noch eine Anmerkung: Der Trend zu immer umfangreicheren Basic-Varianten hat auch zur Folge, daß die einzelnen Dialekte immer mehr auseinanderlaufen. Damit wird das Hauptargument, nämlich dem professionellen Programmierer die in anderen Sprachen üblichen Elemente zur Verfügung zu stellen, ad absurdum geführt. Ein Profi wird nämlich den Teufel tun, ein Programm in einer Sprache zu schreiben, die jegliches Portieren ausschließt. Was ich unter diesen Umständen nun gar nicht verstehe, ist, daß sich GFA bei der Namensvergabe für Schlüsselwörter nicht an den Standard hält, ganz im Gegensatz zu Omikron, die die Namen nebst Semantik so buchstabengetreu bei Microsoft abschreiben, daß nahezu alle PC-Programme unter Omikron-Basic auf Anhieb auf dem Atari ST laufen. So bleibt dann mancher doch bei C, wo es auch schon fast 400 Funktionen gibt, nur mit einem Unterschied: Da erfolgen diese Erweiterungen nach dem ANSI-Standard, weshalb zum Beispiel ein Quicksort-Aufruf in MS-C genauso läuft wie in Turbo-C. Um bei diesem Beispiel zu bleiben: Wohl wissend, daß ein Quick-Sort zum langsamen Bubble-Sort degeneriert, wenn er mit einer schon sortierten Struktur aufgerufen wird, prüft die C-Funktion diese Bedingung ab. In GFA-Basic muß das der Programmierer wissen. Er kann alternativ den auch eingebau-

Die neueste Version von GFA-Basic wurde im Vergleich zum Vorgänger so kräftig aufgebohrt, daß allein die Menge der neu hinzugekommenen Funktionen alle Rekorde schlägt. Lesen Sie, wie unser Tester GFA-Basic 3.0 sieht, nachdem er sich durch diesen Berg hindurchgearbeitet hat.

ten Shell-Sort nehmen, doch der ist (bei unsortierten Daten) drastisch langsamer. Nun aber zum Neuen in GFA-Basic: Man kann die neuen Befehle in vier Gruppen einteilen, nämlich „war nötig“, „ist praktisch“, „ist Luxus“ und „ist exotisch“. Unter „war nötig“ würde ich zuerst die neuen Befehle zur Programmstrukturierung einstuft, wozu SELECT CASE und ELSE IF gehören. Damit lassen sich Mehrfachverzweigungen, wie die in der typischen AES-Event-Schleife endlich so klar programmieren, wie in C oder Pascal. Auch eine neue Strukturmöglichkeit ist ein Schleifenkonstrukt der Form

```
DO UNTIL Bedingung_1 ...
LOOP WHILE Bedingung_2
```

Ob diese „abweisende offene“ Schleife zur Klarheit beiträgt, wage ich allerdings zu bezweifeln. Sicher nötig und ein großer Fortschritt ist die Einführung mehrzeiliger Funktionen, die auch rekursiv aufgerufen werden können.

Bei der Gelegenheit sei eine andere Verbesserung erwähnt, nämlich daß nun Prozeduren auch Var-Parameter kennen und daß jetzt Arrays direkt ohne den (sonst zu programmierenden) Umweg über Zeiger übergeben werden können.

... und etwas Wesentliches fehlt doch

Damit wäre mein Bedarf schon erfüllt, wenn nicht auch hier etwas Wesentliches fehlen würde. Unbestreitbar hat nämlich Programmieren auch mit Datenverarbeitung zu tun, und dazu gehört nun einmal, daß man nicht nur das Programm sondern auch die Daten problemorientiert strukturieren können sollte. Das von Basic angebotene Array als einziges Strukturelement ist da ausgesprochen wenig. Das wohl erkennend hat jetzt auch Microsoft (und

längst nicht als erster) in Quick-Basic 4.0 den Record-Typ von Pascal (bzw. C-Struct) eingeführt. Und wo Microsoft nun endlich diesen alten Zopf des „Fieldings“ mit dem zugehörigen MKx und CVx abgeschnitten hat, bleibt in GFA-Basic diese Krampflosung des „frühen Bill Gates“, die den Umgang mit

Random-Dateien ausgesprochen umständlich macht. Daß diese Lösung aus Kompatibilitätsgründen zu MS-Basic beibehalten wurde, kann ich nicht glauben, denn zahlreiche neue Funktionen des GFA-Basic sind in MS-Basic unter anderem Namen schon bekannt.

Bliebe noch ein Manko zu erwähnen. Die neuen Befehle INSERT und DELETE zum Einfügen oder Löschen von Array-Elementen sind zwar gut gedacht, doch wenn INSERT nur im Rahmen des vorherigen DIM möglich ist, bringt es wenig. Anders ausgedrückt: das „\$DYNAMIC“ von Quick-Basic fehlt.

In der Gruppe „praktisch“, weil die tägliche Arbeit erleichternd, würde ich diese Befehle einordnen: ARRAYFILL zum Initialisieren sowie STORE und RECALL zum schnellen Speichern und Laden von Arrays. Auch hier muß ich aber auch gleich wieder etwas monieren, nämlich daß die beim Omikron-Basic vorhandenen Matrix-Operationen fehlen, auf die man in technisch/wissenschaftlichen Anwendungen nicht verzichten mag. Dieser Kundenkreis scheint aber auch nicht die Zielgruppe zu sein, denn wenn man einmal gewichtet, wo das meiste Neue bei GFA-Basic 3.0 ist, so stößt man auf Schwerpunkte wie Grafik und AES sowie diverse Bit-Operationen. Eine boshafte Bewertung wäre demnach: ideal für Spiele-Programmierer und Bit-Schieber. Doch bleiben wir in der Gruppe „praktisch“, da gibt es noch einiges zu würdigen. So die DEFxxx-Befehle, die erlauben, einer ganzen Gruppe von Variablen mit demselben Anfangsbuchstaben einen Typ zu geben, wozu auch DEFBIT gehört. DEFSGN und DEFDBL werden nicht unterschieden. Die Eingabe ist zwar möglich, sie wird aber vom Editor automatisch durch DEFFLT (Float) ersetzt. Es paßt zwar nicht hierher, steht aber an dieser Stelle im Handbuch: Mit DEFLIST kann das Listing beeinflußt werden, wobei die Schreibweise der Schlüssel-

TEST

worte und ein/kein Typ-Postfix für Variablennamen wählbar sind. In die Gruppe „Luxus“ würde ich die meisten Grafik-Befehle einordnen, weil da viel „doppelt gemoppelt“ ist. Das liegt daran, daß viele GEM-Funktionen zum einen über Basic-Befehle, zum zweiten über die AES-Bibliothek erreichbar sind. Letztere erlauben eine ziemlich C-ähnliche Programmierung. Ähnliche Überschneidungen bestehen zwischen verschiedenen Dateifunktionen und den auch sehr bequem handhabbaren GEMDOS- und BIOS-Funktionen.

Ein Beispiel für „exotisch“ ist VOID. Wie in C heißt das hier, daß der Wert einer Funktion nicht interessiert. Nun kann man aber anstatt VOID auch die Tilde (~) schreiben. Der Unterschied: Bei der Tilde wird ein Integer-Ausdruck berechnet und dann vergessen, bei VOID wird der Integerausdruck noch in eine Fließkommazahl umgewandelt und diese dann ignoriert. Den Abschnitt habe ich im Handbuch zweimal gelesen und immer noch nicht verstanden. Wenn mir also jemand den Sinn der Übung erklären kann...

Editor optimiert

Der Editor sieht auf den ersten Blick aus, wie der der Version 2.0, es gibt jedoch einige bemerkenswerte Unterschiede. So kann jetzt über das Atari-Symbol in ein GEM-Menü umgeschaltet werden, das den Aufruf von Accessories erlaubt. Rechts oben läuft immer eine Uhr mit, darunter wird die aktuelle Zeilennummer angezeigt. Prozeduren können mit der Help-Taste auf die Kopfzeile reduziert und mit einem erneuten Druck auf die Help-Taste wieder auf die volle Länge gebracht werden. Dadurch ist ein Listing wesentlich klarer lesbar. Die Undo-Taste tut jetzt, was der Name verspricht, nämlich eine geänderte Zeile wieder herzustellen, jedenfalls solange die Zei-

```
Defint "A-Z"
Size%=8190
Dim Flags%(Size%)
Print Times$
Count%=0
For I%=0 To Size%
  Flags%(I%)=1
Next I%
For I%=0 To 8190
  If Flags%(I%) Then
    Prime%=I%+I%+3
    K%=I%+Prime%
    While K%<=8190
      Flags%(K%)=0
      K%=K%+Prime%
    Wend
    Count%=Count%+1
  Endif
Next I%
Print Count$
Print Times$
```

Bild 1. Der Sieb-Test in Basic so geschrieben, daß er in nahezu allen Dialekten läuft

```
DEFINT "A-Z"
PRINT TIMES$
PRINT @ack(3,4)
PRINT TIMES$
END
FUNCTION ack(m,n)
  IF m=0 THEN
    RETURN n+1
  ELSE
    IF n=0 THEN
      RETURN @ack(m-1,1)
    ELSE
      RETURN @ack(m-1,@ack(m,n-1))
    ENDIF
  ENDIF
ENDFUNC
```

Bild 2. Die Ackermann-Funktion ist ein wesentlich härterer Test der Rekursionsfähigkeit als das übliche n!

```
10 DEFINT "A-Z"
20 PRINT TIMES$
30 PRINT FN Ack%(3,4)
40 PRINT TIMES$
50 END
60 DEF FN Ack%(M%,N%)
70 IF M%=0 THEN RETURN N%+1 ELSE IF N%=0 THEN RETURN FN Ack%(M%-1,1) ELSE RETURN FN Ack%(M%-1,FN Ack%(M%,N%-1))
```

Bild 3. Hier geht es um Zeile 70. Bei so komplexen Befehlsfolgen ist GFA-Basic überfordert

le noch nicht verlassen wurde. Der Editor selbst ist mit einem Syntax-Prüfer ausgerüstet und übernimmt auch die Formatierung und Strukturierung des Listings.

Zu bemängeln ist, daß die Blockauswahl nicht per Maus-Dragging möglich ist, ein Feature, daß sich inzwischen sogar in den sonst so tastaturorientierten Profi-Kreisen durchgesetzt hat (z. B. bei Tempus 2.0, Megamax-C, Turbo-C). Beim Thema Block sind auch so zwei bis drei Fehler anzumerken. Wählt man „Block drucken“, wird in der Menü-Leiste „Quit“ aufgehellt, reagiert aber nicht. Wählt man dann „Nein“ in der Dialog-Box, wird trotzdem eine Seite Papier vorgeschoben. Ist der Drucker nicht bereit, muß man sich trotz „Nein“ für 30 Sekunden (Wartezeit des GEMDOS) gedulden. Die Fehler treten nach dem Start manchmal

auf; wenn man erst einen Block gedruckt hat, immer. Auch in diese Rubrik fällt, daß im Listing das totale Chaos ausbricht, wenn man einen Block als Teil einer Prozedur drucken will, die Kopfzeile der Prozedur aber nicht mit ausgewählt hat.

Diese Punkte halte ich schon für kritisch, denn niemand wird bei größeren Programmen immer das ganze Listing drucken, sondern sich auf den Änderungsbereich beschränken. Das Handbuch ist entsprechend dem Sprachumfang kräftig gewachsen und hat auch deutlich an Qualität gewonnen. Im Gegensatz zum Vorgänger, bei dem ja, um das Kopieren zu verhindern, schwarz auf rot gedruckt wurde, hat man hier eine andere Schutzmethode gewählt: Das Handbuch ist so umfangreich, daß es billiger ist, sich den Interpreter zu kaufen, als dieses

```
ALERT 1 "Disk in A: einlegen", 1, "Ok", ax
ss=SPACES(512)
DO
  xx=xBIOS(0, L:UARPTR(ss), L:0, 0, 1, 0, 0, 1)
  IF xx=0 THEN
    PRINT "Fehler "xx;" Taste drücken (ESC=Abbruch)"
    INP(2)=27
  ENDIF
LOOP
ASC(ss)=4H60
PRINT "Diskette infiziert oder bootfähig"
ALERT 3, "Diskette infiziert oder bootfähig", 1, "Löschen/Weiter", xx
IF xx=1 THEN
  MID$(ss, 1)=STRING$(6, 0)
  MID$(ss, 55)=STRING$(512, 4HES)
  VOID xBIOS(18, L:UARPTR(ss), L:-1, -1, 0)
  REPEAT
    xx=xBIOS(9, L:UARPTR(ss), L:0, 0, 1, 0, 0, 1)
  UNTIL xx=0
  PRINT "Schreibfehler "xx;" Taste drücken (ESC=Abbruch)"
  INP(2)=27
ENDIF
ELSE
  PRINT "Disk sieht normal aus"
ENDIF
LOOP
```

Bild 4. So sieht die neue Befehlszeile von GFA-Basic 3.0 aus. Das Programm dient zum Entfernen von Virus-Programmen aus dem Bootsektor

TEST

Volumen zu kopieren. Das Handbuch selbst ist primär ein Referenz-Manual, das für jedes Schlüsselwort (oder eine logische Gruppe) eine Seite mit Syntax, Semantik und einem kurzen Beispiel bietet. Wer mehr wissen will, kann sich das Buch zu GFA-Basic 3.0 (von GFA) zusätzlich kaufen. Dieses Buch kann Einsteigern nur dringend empfohlen werden. Der erfahrene GFA-Basic-Programmierer dürfte jedoch anhand des zum Lieferumfang gehörenden Manuals auch mit den neuen Funktionen problemlos zurechtkommen.

Schneller und genauer

Um einmal Tempo und Rechengenauigkeit gleichzeitig zu messen, habe ich einfach die Konstante Pi 10000mal auf eine mit Null initialisierte Variable addiert und dann davon 10000mal Pi wieder subtrahiert. Nach Adam Riese müßte das Ergebnis wieder

Null sein, die folgende Tabelle zeigt die Ergebnisse:

Version	Restfehler	Zeitbedarf
GFA 2.0	$-3.9 \cdot 10^{-8}$	6 s
GFA 3.0	$-1.8 \cdot 10^{-10}$	4 s
Omikron	$-1.4 \cdot 10^{-15}$	4 s

Sie sehen, GFA-Basic 3.0 hat sich gegenüber dem Vorgänger gebessert, kommt aber an Omikron bei weitem nicht heran, kann es auch nicht, da Omikron mit 19 anstatt 12 signifikanten Stellen rechnet. Trotzdem ist Omikron, wenn man die höhere Rechengenauigkeit würdigt, sogar schneller. Bei Integers sleht der Vergleich unterschiedlich aus. Im Sieb-Test (*Bild 1*) braucht GFA-Basic 3.0 10 Sekunden, Omikron-Basic hingegen 12 Sekunden. Bei hochrekursiven Funktionen wie der Ackermann-Funktion (*Bild 2*) sind beide mit 8 Sekunden

den exakt gleich schnell. So ein Konstrukt wie die Zeile 70 von *Bild 3* ist nur in Omikron-Basic möglich – aber auch da nicht nötig. Wie das nachher compiliert aussieht (der Compiler zum GFA-Basic 3.0 ist angekündigt) bleibt abzuwarten. Ich kann aber schon einmal Ziele setzen. Auf einem 286er-PC mit 10 MHz Takt (PS/2-50) sind Turbo-Basic und Quick-Basic in diesen Benchmarks rund 25mal schneller.

Für GFA-Freaks empfehlenswert

Zusammenfassend kann man die Version 3.0 so beurteilen: Wer bisher mit GFA-Basic gearbeitet hat und dabei bleiben will, sollte auf jeden Fall „upgraden“, da lohnt es sich. Wer mit Omikron-Basic arbeitet, sollte nicht wechseln. Die Neuheiten der Version 3.0 wiegen die Nachteile eines Umstiegs bei weitem nicht auf. □

Die Entwicklung von hochintegrierten Chips für XT-, AT- und 386-Maschinen ist noch lange nicht abgeschlossen. Zu den seit längerer Zeit im PC-Chip-Markt agierenden Herstellern wie z. B. Zymos und Chips & Technologies hat sich G-2, eine Tochterfirma der kalifornischen ASIC-Schmiede LSI-Logic, hinzugesellt. Anfang April stellte G-2 einen Chip für zum Modell 30 der PS/2-Serie kompatible Computer und für XT-Clones vor, sowie einen VGA-Chip und einen Chipsatz für 80386-Maschinen.

Der Modell 30/XT-Chip, genannt GC100, arbeitet bis zu einer Taktfrequenz von 10 MHz. Ein Modell-30-Clone ist damit fast bis zu 25 % schneller als das Originalprodukt, dessen Taktfrequenz nur 8 MHz beträgt. Je nach verwendeter CPU (8088 oder 8086) und Speicherbausteinen kann der Entwickler den integrierten Wait-State-Generator programmieren. Mit dem GC100 lassen sich die CPU- und Zusatzfunktionen eines Modell-30-Clones auf nur 14 ICs reduzieren. Bei einem Super-XT verringert sich der Bauteileaufwand von 55 auf 12 Chips (ohne Speicher).

Nur aus einem Peripherie-Controller, einem CPU- und Speicher-Controller und dem Bus Bridge Interface genannten Baustein besteht der 80386-Chipsatz. Mit ihm kann ein PC-Hersteller AT-kompatible Computer entwickeln, die außer den Speicherchips nur noch zwölf weitere ICs beinhalten.

Bis zu 24 MByte darf der Speicher bei Verwendung des 80386-Chipsatzes groß sein. Die jeweilige Gerätekonfiguration läßt

Neue Chips für die PC-Welt

sich in einem EEPROM (elektrisch löscht- und programmierbarer ROM) speichern, so daß beim Booten des Systems oder auch während des Betriebs sich die Systemparameter ändern lassen, um die Leistungsfähigkeit des Systems zu optimieren. Vom Hersteller können die Parameter der Speicherverwaltung, des Speichertyps, des Coprozessors und der Ein-/Ausgabe programmiert werden.

Folgende Parameter der Speicherverwaltung sind einstellbar:

- Page Mode
- Interleaving
- Anzahl der Wait States bei RAM- und ROM-Zugriffen
- Refresh Rate
- Shadow-RAM.

An Speichertypen werden dynamische und statische RAMs als auch EPROMs und EEPROMs unterstützt. Auch bei den Coprozessoren ist der Chipsatz flexibel verwendbar: Der Entwickler kann zwischen dem 80287 und dem 80387 wählen.

Der neue VGA-Chip von G-2 wird zukünftigen Grafikkarten Beine machen. Sämtliche VGA-Funktionen wurden optimiert und in dem GC205 genannten Baustein untergebracht. Durch die Einbindung der von IBM nicht dokumentierten Register soll der Chip auch zu zukünftigen VGA-Produkten

kompatibel sein. Er ist bereits für Pixeltaktfrequenzen bis zu 65 MHz und für Video-RAMs (VRAMs) ausgelegt. Um eine hohe Bildverarbeitungsgeschwindigkeit zu erreichen, wurde die Zahl der Wartezyklen bei Speicherzugriffen drastisch reduziert. Während die Original-VGA nur in jedem siebten Zyklus einen Zugriff auf den Videospeicher zuläßt, ermöglicht der GC205 in Verbindung mit VRAMs bei allen Standard-VGA-Betriebsarten ein Verhältnis von 1:1. Bei einer Auflösung von 800 × 600 Bildpunkten beträgt das Verhältnis 1:2 und ein Wert von 1:4 wird bei einer Auflösung von 1024 × 768 erreicht. Einen weiteren Geschwindigkeitszuwachs wird mit der 16-Bit-Schnittstelle zum E/A-Bus erzielt. Für einfache Anwendungen kann man den Baustein auch in die Betriebsart mit 8-Bit-Schnittstelle bringen.

Mit dem Hardware-Cursor wird das Erscheinungsbild von grafischen Benutzeroberflächen wie z. B. Microsoft Windows verbessert. Viele VGA-Chips können nur einen per Software generierten Mauszeiger unterstützen, der flimmert und sich nur langsam über den Bildschirm bewegt, da er bei jeder Cursorbewegung vom Programm aktualisiert werden muß. Der Hardware-Cursor des GC205 ist flimmerfrei und läßt sich schnell über den Bildschirm bewegen. In Verbindung mit VRAMs kann der Chip bei einer Auflösung von 720 × 540 Pixeln 256 Farben gleichzeitig darstellen; bei 1024 × 768 Bildpunkten sind es immerhin noch 16 Farben. Außerdem ist er kompatibel zu EGA, CGA- und MDA.

Nach Unterlagen von G-2

Günther Sternberg

Die Norton-Guides

Peter Norton ist seit mehreren Jahren bekannt für seine guten und nützlichen Utilities, sowie für eine Reihe von Büchern über das Innenleben und die Programmierung von Personal Computern. Sein bekanntestes Produkt, die Norton-Utilities, ist weltweit der Renner unter den Hilfsprogrammen für PCs. Mit den Norton-Guides, im Vertrieb von H+B EDV in Tettnang, erreichte uns eines seiner neuesten Produkte. Mit den Norton-Guides sind die für einen PC-Benutzer am häufigsten benötigten Informationen zu einem bestimmten Thema (z. B. zu einer bestimmten Programmiersprache) jederzeit im Computer auf Abruf verfügbar. Der Anwender soll bei Bedarf möglichst schnell und mit wenig Aufwand die benötigte Hilfestellung erhalten. Ein einfaches und sehr leicht zu bedienendes Datenbank-Programm bildet den Kern der Norton-Guides. Das eigentliche „Handbuch“ zu einem bestimmten Thema kauft man sich dann in Form einer passenden Datenbank dazu. Für den Test standen mir die Datenbanken für folgende Programmiersprachen zur Verfügung:

- Turbo-Pascal
- Turbo-Basic
- Turbo-C
- Microsoft-Macro-Assembler
- Microsoft-C
- Basica (IBM-Basic)
- Quick-Basic

Schnelle Installation und einfache Handhabung

Zum Aufbau des Informationssystems benötigt man einen IBM-PC/XT/AT oder Kompatiblen mit Festplatte und MS-DOS-Betriebssystem, das Norton-Guide-Datenbank-Programm und eine oder mehrere der Datenbanken. Die Installation ist denkbar einfach. In ein beliebiges Subdirectory kopiert man das Datenbank-Programm mit dem Namen NG.EXE und von der Datenbank-Diskette die Datei SETUP_??.EXE (anstelle von ?? steht der Name der jeweiligen Datenbank). Letztere Datei enthält die Datenbank in komprimierter Form. Der Aufruf SETUP_?? legt die eigentliche Datenbank an und die Datei SETUP_??.EXE kann gelöscht werden.

Wer kennt folgende Situation nicht: Man arbeitet am Computer und die Syntax eines Befehls oder die richtige Option fällt einem nicht ein. Natürlich sind die Handbücher auch nicht zur Hand. Mit dem neuesten Produkt von Peter Norton, den Norton-Guides, packt man die Handbücher in den Computer.

Damit man nach Möglichkeit zu jedem Zeitpunkt an die Informationen der Norton-Guides herankommt, lädt man das Programm am besten beim Booten als Speicherresidentes Programm. Die Norton-Guides belegen ungefähr 65 KByte Hauptspeicher, unabhängig von Anzahl und Größe der vorhandenen Datenbanken. Wer sich seinen Speicher nicht gerne mit allen möglichen residenten Programmen vollstopft, der kann die Norton-Guides auch nur für eine einzelne Auskunft oder für den Zeitraum der Bearbeitung des nächsten Befehls oder Programms laden. In jedem Fall erfolgt die Aktivierung der geladenen Norton-Guides durch Betätigung einer Tastenkombination (Standard: Shift-F1). Der Aufruf der Norton-Guides während der Arbeit mit Open Access II, Turbo-Pascal, Wordstar und noch einigen anderen Programmen klappte einwandfrei. Lediglich Euroscript Version 2 verträgt sich mit den Norton-Guides nicht. Hier kam es jedesmal zu einem Systemabsturz, da Euroscript die Tastatur direkt bedient.

Hoher Informationswert der Datenbanken

Zuerst stand ich den Norton-Guides sehr skeptisch gegenüber. Im Verlauf meiner inzwischen über 10jährigen Tätigkeit als Software-Entwickler sind mir schon eine ganze Reihe von „online“-Hilfen untergekommen. Aber alle Hilfestellungen im Computer, auch an Großrechnern, wo Speicherplatz wirklich keine Rolle spielt, waren für mich nur selten wirklich von Nutzen. Die meisten dieser Hilfsprogramme arbeiten nach dem folgenden Schema: Man gibt ein Stichwort ein, zum Beispiel den Namen eines Kommandos, und erhält dann zwei oder drei Zeilen zur Syntax und eine kleine Kurzbeschreibung. Meist ist die erhaltene Information so spärlich und ungenau, daß einem der Griff zum Handbuch doch nicht

erspart bleibt. Zugegeben, als professioneller Programmierer stelle ich viel höhere Ansprüche an ein solches Informationssystem als ein Anfänger, dem zum Beispiel einfach nur die Syntax des Print-Befehls in Basic entfallen ist. Aber selbst meinen hohen Ansprüchen in Bezug auf Genauigkeit und In-

formationstiefe wurden die Norton Guides voll gerecht. Mit Freude nahm ich das Vorhandensein einer Vielzahl von Tabellen und Informationen zur Kenntnis, wegen denen ich schon oft zum Handbuch greifen mußte. Man erkennt sehr deutlich, daß Profis bei der Auswahl der Themenkreise am Werk waren. Der hohe Informationsgehalt spiegelt sich auch in der Größe der Datenbanken wieder. So ist die kleinste Datenbank (Basica) ungefähr 230 KByte und die größte Datenbank (Assembler) fast 600 KByte groß. Der einzige Wermutstropfen für einige Anwender dürfte die Tatsache sein, daß das Handbuch, das Datenbank-Programm und die Datenbanken selbst bisher nur in englischer Sprache vorliegen. Eine deutsche Version dürfte aber wohl nur eine Frage der Zeit sein.

Angenommen Sie sind gerade mit der Erstellung eines Turbo-Pascal-Programms beschäftigt und Ihnen fällt die Reihenfolge der Parameter bei dem „Intr“-Befehl nicht ein. Dann betätigen Sie einfach den Hotkey (z. B. Shift-F1). Vielleicht erhalten Sie dann auf Ihrem Monitor ein Bild, daß der in Bild 1 gezeigten Hardcopy entspricht. Beim Aufruf sucht das Datenbank-Programm automatisch in der zuletzt verwendeten Datenbank nach dem in der Nähe des Cursors befindlichen Stichwort. In unserem Beispiel stand der Cursor zuletzt bei dem Wort „Intr“ des Pascal-Programms. Entsprechend steht der Cursor nun in der Zeile mit der Kurzinformation zum „Intr“-Befehl. Durch die Betätigung der Enter-Taste erhält man die ausführliche Information (Bild 2). Selbstverständlich kann man auch die Datenbank wechseln oder gezielt per Stichwort nach Informationen suchen. Falls man kein Stichwort parat hat, kann man in der Datenbank „blättern“.

Ein weiteres Beispiel soll die Vielfalt an Informationen aufzeigen. Die Datenbank zum Assembler enthält folgende Einzelgebiete:

TEST

○ Assembler allgemein

- Instruction Set
- ROM Bios
- Low Memory Usage
- Flags Register
- Addressing Modes
- Effective Address
- System ID Byte

○ MASM im Detail

- Pseudo Ops
- Operator Precedence
- Reserved Words

○ DOS-Betriebssystem

- Funktionen
- Interrupts
- Error Codes
- File Attributs
- Standard Handles
- PSP Description
- FCB Fields

○ Tabellen

- ASCII Chart
- Line-Drawing Chars
- Special Characters
- Color Chart
- Keyboard Codes

Eine ganze Reihe der hier nur stichwortartig aufgeführten Punkte findet man nicht einmal im Handbuch zum Microsoft-Makro-Assembler. Hinter den Überschriften verstecken sich keineswegs nur lapidare Kurzinformationen. Bild 3 stellt nur ungefähr ein Drittel der Erklärung zum Punkt „FCB Fields“ dar, einem nun wirklich sehr speziellen Detail von MS-DOS.

Eigene Datenbanken erlaubt

Falls man mit den fertigen Datenbanken nicht zufrieden ist oder man sich einfach nur eine eigene, maßgeschneiderte Datenbank anlegen will, sind die Norton-Guides trotzdem die richtige Wahl. Das Datenbank-Programm ist auch ohne Datenbanken erhältlich und versetzt einen im Umgang mit Datenbanken unerfahrenen Benutzer in kurzer Zeit in die Lage, sich sein eigenes Informationssystem aufzubauen. Man benötigt lediglich noch ein beliebiges Textverarbeitungsprogramm. Der Aufbau einer Datenbank ist so einfach, daß zehn Seiten des insgesamt 50seitigen Handbuchs ausreichen, um alles Wichtige zu erklären.

Die Norton-Guides sind keineswegs nur als Erste Hilfe für den Anfänger aufzufassen, sondern selbst für einen fortgeschrittenen Anwender eine beachtliche Informationsquelle. □

```
Line 10 Col 3 Insert Indent D:DEMO.PAS
Var Regs : Record
    Case Boolean of
        True : (AX, BX, CX, DX, BP, SI, DI, DS, ES, Flags : Integer);
        false: (AL, AH, BL, BH, CL, CH, DL, DH : Byte)
    End;
```

```
Begin
    Regs.AH := 0;
    Regs.AL := 4;
Intr
```

Expand	Search...	Turbo Pascal » Pascal » Turbo Pascal » ...	Tables
Insert		Inserts a Substring into a String	proc
InsLine		Inserts a Line on the Screen	proc
Int		Returns the Integral Portion of a Value	func
Integer		Numbers from -32768 to 32767	type
Intr		Calls System Software Interrupt	proc
IOResult		Returns I/O Error Status	func
KBD		Predefined File Assigned to Keyboard	file
Keypress		Returns TRUE if Key Has Been Pressed	func

Bild 1. Die Norton-Guides melden sich so, daß man die zuletzt bearbeitete Stelle noch sieht

See also: MSDos		Turbo Pascal » Pascal » Turbo Pascal » ...
Intr	Calls System Software Interrupt	
Intr(IntNum : Integer; var Parm : Registers);	[TP]	
Calls the system software interrupt routine IntNum. Parm is a variable of type Registers (defined below).		
IntNum	ID number of software interrupt	
Parm	Special record of type Registers	
Notes:	Registers is a specific record data type that you must define in your program. It takes the following format:	
type Registers = record case Boolean of False: (AX, BX, CX, DX, BP, DI, SI, DS, ES, Flags: Integer); True : (AL, AH, BL, BH, CL, CH, DL, DH : Integer) end;		

Bild 2. Ein Ausschnitt der Erklärungen zum Intr-Befehl von Turbo-Pascal

Assembly Language » DOS » FCB Fields

See also: File attributes

File Control Block (FCB) Structure

Offset	Length	Field	Initialized to	Initialized by
00h	1	Drive code	Drive specified	User
01h	8	Filename	Filename	User
09h	3	Extension	Filename extension	User
0Ch	2	Current Block	00h	DOS
0Eh	2	Record Size	80h	DOS; see below
10h	4	File Size	Value in directory	DOS
14h	2	Date	Value in directory	DOS
16h	2	Time	Value in directory	DOS
18h	8	Reserved		
20h	1	Current Record	See notes below	User
21h	4	Random Record Number	See notes below	User

An extended File Control Block is used to access files with special attributes. An extended File Control Block has three additional fields, starting at offset -07h (minus 7), as follows:

Bild 3. Eine weitere Betätigung der Enter-Taste liefert die Erklärungen des Querverweises (File Attributes)

Peter Wollschlaeger

Turbo-C für den Atari ST

Was vorher schon in der Gerüchteküche so durchgesickert war, bekommt man schriftlich (Bild 1), wenn man das „About“ aufruft. „About“ kennen Sie nicht? Nun, so heißt das Desk-Info beim Macintosh, und auch sonst ist einiges an diesem Programm gute alte Macintosh-Tradition, und damit wären wir beim Thema. Borland ist zur Zeit damit beschäftigt, Turbo-Pascal vom Macintosh auf den Atari ST zu portieren. Als nun Heimsoeth mit einem ST-C von SoftDesign München kam, und dafür den schon geschützten Namen Turbo-C haben wollte, gab es erst einmal Wirbel im Konzern. Doch das Produkt überzeugte und wurde deshalb in Borlands ST-Linie eingepaßt, und die ist vom Macintosh geprägt. Die Zielvorstellung heißt: Source eintippen, Run aus dem Menü wählen und Sekunden später müssen Compiler und Linker fertig und das Programm gestartet sein.

Integrierte Entwicklungsumgebung

Tatsächlich funktioniert dieses schon von Turbo-Pascal her bekannte Prinzip auch bei Turbo-C für den Atari ST. Editor, Compiler und Linker stehen als integriertes System im Speicher, nur die in C unvermeidlichen Libraries werden noch gelinkt. Damit letzteres den Fluß nicht bremst, sollte man allerdings eine Festplatte oder genügend Platz für eine RAM-Disk haben. Für Leute mit Platzproblemen oder notorische Tipper gibt es sogar eine Kommando-Version, doch der echte ST-Freak wird wohl TC.PRJ anklippen, und damit in der Oberfläche von Bild 2 landen. Die Menüs sind zwar selbst erklärend, aber doch einige Kommentare wert. „Open“ läßt die übliche File-Select-Box erscheinen, jedoch dann nur Dateien mit der vorgewählten Erweiterung. Wichtiger: Man darf bis zu sechsmal diesen Punkt aufrufen, also bis zu sechs Fenster gleichzeitig öffnen. Das Directory-Menü bietet die wesentlichen Operationen des Desktop an, die Accessories stehen auch zur Verfügung. Will da etwa noch jemand die Turbo-C-Umgebung verlassen? Zum Edit-Menü wäre anzumerken: Cut, Copy und Paste funktionieren präzise so, wie beim Macintosh. Auch die Textauswahl

Es gibt schon ein halbes Dutzend C-Compiler für den Atari ST, und einige sind wirklich gut. Wenn in dieser Situation Heimsoeth und Borland in den Markt einsteigen, dann lohnt es sogar, sich die Vorabversion anzusehen. Bis zu sechs Dateien kann man gleichzeitig bearbeiten und compilieren.



Bild 1. Gleich drei „Väter“ nennt das „About Turbo-C“

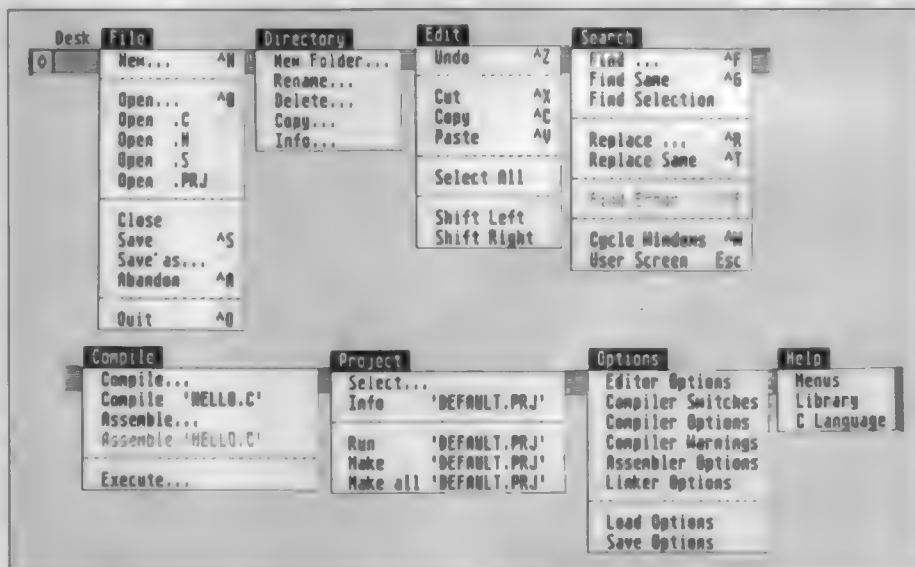


Bild 2. Reiche Auswahl: Alle Menüs des Editors

geschieht per Maus-Dragging. Die Auswahl wird auch schwarz unterlegt und sogar die Tastenkürzel entsprechen genau den Apple-Vorschriften. Das Search-Menü bietet alles zum Suchen und Ersetzen und noch ein paar Bonbons. „Find Error“ bedeutet: Der Compiler schreibt alle Fehler in ein eigenes Fenster. Nun reicht es, dort auf einen Fehler zu klicken, Ctrl-E zu tippen (oder das Menü anzuwählen) und schon ist man auf der entsprechenden Zeile im Quelltext. Das Umschalten zwischen den Fenstern kann mit der Maus erfolgen, solange man nicht

ein Fenster völlig abdeckt. In diesem Fall hilft „Cycle Windows“. Der „User Screen“ ist der Ausgabe-Schirm der Programme, die nie die Windows von Turbo-C überschreiben können. Mit diesem Menü-Punkt oder schneller mit der Escape-Taste kann man zwischen beiden hin- und herschalten. Das Compile-Menü gestattet den Aufruf des Compilers und des Assemblers. Letzterer ist übrigens auch nicht schlecht, soweit ich das ohne Dokumentation beurteilen kann. Auf jeden Fall kann er auch Code für den 68020 und 68030 und die Koprozessoren erzeugen. Bild 3 zeigt die Optionen, die man aus dem Editor heraus vorwählen kann. „Execute“ erlaubt den Aufruf anderer Programme. Bei TOS-Programmen klappt das auch, bei GEM-An-

wendungen führte das (zumindest in der von mir getesteten Vorabversion) zum totalen System-Crash.

Viele Features

Unter „Project“ verbirgt sich eine sehr komfortable Make-Utility. Dazu muß man lediglich in einer Textdatei alle Source-, Objekt- und Library-Dateien eintragen, die man so hat, und dann „Make Project“ oder „Run Project“ aufrufen, wenn das Programm auch gleich laufen soll. „DE-

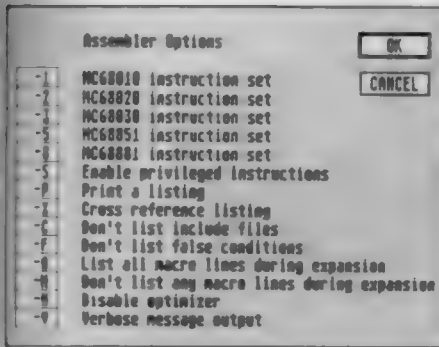


Bild 3. Ein zukunftsicherer Assembler wird mitgeliefert

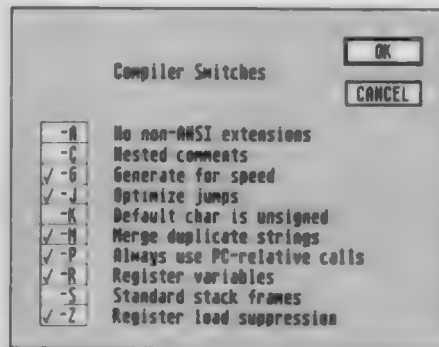


Bild 4. Compiler-Switches per Maus einstellbar, Optimizer inklusive

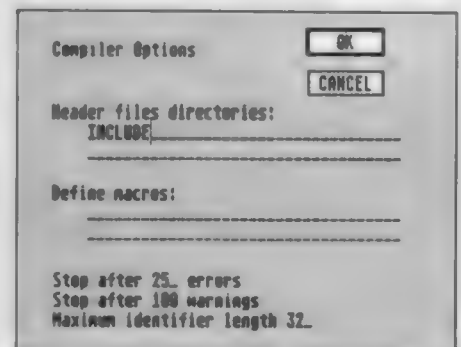


Bild 5. Der Programmierer kann selbst bestimmen, wann der Compiler aufgeben soll

FAULT.PRJ" steht immer zur Verfügung, wenn man nicht via „Select“ ein anderes Projekt gewählt hat. In diesem Fall wird nur die Standard-Library eingebunden. Wenn man auf Systemroutinen zugreifen will, muß man die TOS- und/oder die GEM-Library einbinden. Ich habe mir einmal die zugehörigen Header-Dateien angesehen und konnte auf Anhieb keine Funktion feststellen, die darin fehlte. Was sich unter den Optionen verbirgt, zeigen die *Bilder 3 bis 6*. Sie sehen anhand von *Bild 4*, wie vielseitig der Compiler ist. Sogar der sonst separat zu bedienende (und zu bezahlende) Optimierer ist schon eingebaut. Bei *Bild 5* würden Sie bitte, daß Sie nun selbst vorgeben können, bei welcher Fehler- und Warnungszahl der Compiler abbricht. Die Fehlermeldungen bieten übrigens sehr hohen Komfort. So werden zum Beispiel unbenutzte und sogar deklarierte aber nicht initialisierte Variable gemeldet. Die Typprüfung ist sehr streng. Da muß man entweder sauber programmieren oder gute Nerven beim Ignorieren der Warnungen haben. Zum Linker (und *Bild 6*) wäre anzumerken, daß die Symboittabelle zwar von anderen Debuggern verstanden wird, ein Debugger aber (hoffentlich *noch*) nicht zum Lieferumfang gehört. Sie haben es schon erkannt, der Linker verarbeitet DR-Format. Es ist zwar die unfeine Art, mit Vorabversionen Benchmarks zu fahren, aber wenn das Ergebnis nicht schlecht ist, darf man das wohl. Im Sieb-Test laut *Bild 7* wurden zwischen den beiden „getchar()“ 10,0 Sekunden gestoppt. Megamax-C brauchte dafür 12,8 Sekunden. Daß der Compiler auch in anderen Benchmarks glänzen wird, ist zu erwarten, wenn man sich den erzeugten Code anschaut. Ausdrücke werden sehr effektiv übersetzt und die Parameterübergabe läuft nicht wie in C sonst üblich über den Stack, sondern nimmt den schnelleren Weg über Register. Der Editor läßt sich wahlweise mit den Cursor-Tasten oder mit der Maus bedienen. Beim Scrollen mit den Cursor-Tasten ist er nicht besonders schnell, es gibt aber auch kein Nachlaufen, und das seitenweise Umblättern ist wiederum sehr flott. Beanstanden würde ich, das es recht

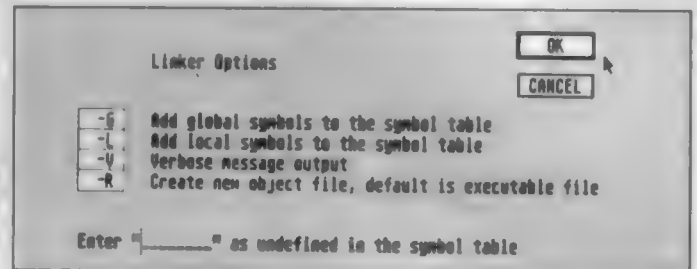


Bild 6. Der Linker hält sich an das DR-Format

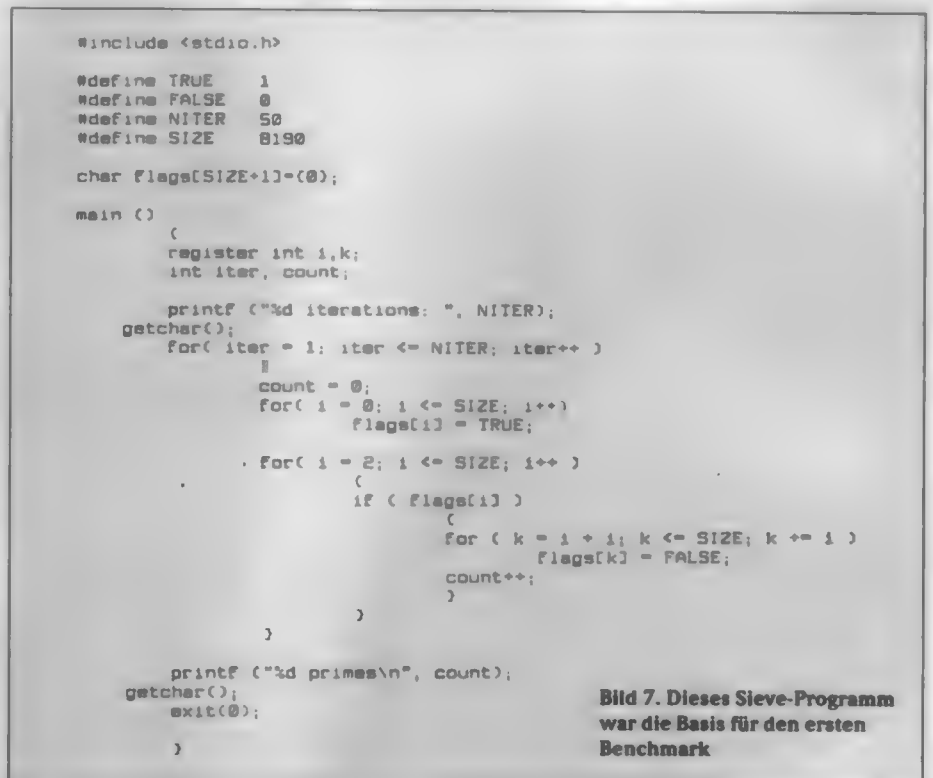


Bild 7. Dieses Sieve-Programm war die Basis für den ersten Benchmark

lange dauert (Taste ist kräftig zu drücken und nicht nur zu klicken), bis man den Cursor mit der Maus gesetzt hat. Der Compiler und der Linker sind sehr schnell. Setzt man eine Festplatte oder eine RAM-Disk (für die Libraries) ein, so laufen auch mittelprächtige Programme in Sekunden: Turbo-C wird seinem Namen voll gerecht. Da es sich hier wirklich um ein integriertes System handelt, und nicht nur um Einzelprogramme, die unter einer Shell liegend nacheinander aufgerufen werden, schlägt es die Mitbewerber eindeutig in den Turn-Around-Zeiten. Das Fazit ist erfreulich, und

das gleich in dreifacher Hinsicht. Zuerst scheint das Produkt Turbo-C doch sehr gelungen zu sein. Der Vorbehalt ist nötig, weil wir mit Vorabversionen keine tiefergehenden Vergleichstests fahren können. Zum zweiten ist positiv zu sehen, daß mit Turbo-C wieder Bewegung in den Markt kommt. Die Konkurrenz muß sich jetzt etwas einfallen lassen. Last not least können die Atari ST-Freaks erfreut sein. Daß sich jetzt auch ein so großes Software-Haus wie Heimsoeth/Borland um den Atari ST bemüht, beweist, daß sich dieser Computer durchgesetzt hat. □

Christian Strasheim

C mit Lichtgeschwindigkeit

Das neue Megamax-C-Entwicklungssystem für den Atari ST

Der Profi mußte bisher auf alles verzichten, was zur effizienten Erstellung größerer Programme nötig ist, beispielsweise ein wirkliches MAKE oder kurze Turn-Around-Zeiten. Das wird jetzt (bald) ein Ende haben.

Die Lösung aller Probleme heißt Megamax C 2.0 oder besser: Laser C. So nennt die Firma Megamax ihr neuestes Produkt, das seit kurzem auf dem amerikanischen Markt erhältlich ist. Und in der Tat ist der neue Name voll gerechtfertigt, denn Laser C hat mit dem alten Megamax C nur noch wenig gemein.

Eine neue Shell

Die Installation des Compilers auf der Festplatte vollzieht sich ebenso einfach wie beim alten Megamax-System: Einfach alles in einen Ordner namens MEGAMAX, die Include-Dateien in ein Subdirectory HEADERS und die Shell ins Rootdirectory kopieren. Nach einem Doppelklick auf LASER.PRГ – so der Name der neuen Shell – präsentiert sich dem Anwender eine wesentlich erweiterte Menüleiste (Bild 1). Denn die Shell ist nicht mehr nur Shell, sondern Shell und Editor in einem. Der Vorteil dieses Konzepts liegt auf der Hand: Drastische Verkürzung der Entwicklungsdauer von Programmen. Aber nicht genug damit. Der Compiler, der Linker und Make (jetzt ein eigenständiges Programm) lassen sich automatisch als RAM-residente Programme laden. Infolgedessen erhöht sich die Geschwindigkeit beim Compilieren und Linken gegenüber Megamax 1.0 noch einmal fast um den Faktor 2; damit ist Laser C dem Lattice-C-System beim Compilieren und Linken um den Faktor 4 überlegen. Man sollte es nicht glauben, aber es geht noch schneller.

Hat man genügend Speicher zur Verfügung, lassen sich effektiv alle Zugriffe auf Festplatte oder Diskette „wegrationalisieren“. Das Geheimnis ist ein in die Shell integrierter Cache. Selbstverständlich kann man diesen auch deaktivieren, also bestimmten Pro-

Megamax-C hat sich in Deutschland – ganz anders als in den USA – als C-Compiler für den Atari ST weitgehend durchgesetzt. Es ist seinen Konkurrenten durch Kompaktheit und Geschwindigkeit sowohl des Compilers als auch des erzeugten Codes weit überlegen und ist einfach zu bedienen.

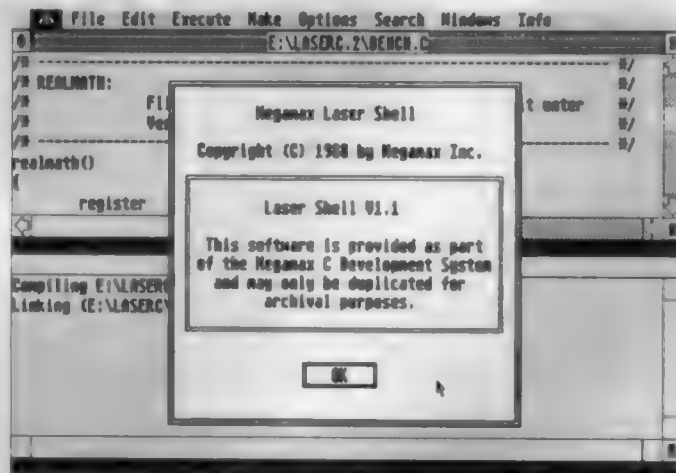


Bild 1. So präsentiert sich das neue Megamax C

grammen ein direktes „Durchschreiben“ auf die Massenspeicher erlauben. Das umständliche Gefummel mit einer RAM-Disk (am Anfang alles von der Diskette in die RAM-Disk kopieren, am Ende die geänderten Sachen sichern) entfällt somit. Bei 1 MByte RAM läßt sich auch mit einer Diskette vernünftig arbeiten – solange es sich um kleinere Programme handelt.

Editierisches

Der integrierte Editor entspricht dem Editor des alten Systems, die Menüeinträge sind identisch. Eine angenehme Überraschung ist die merklich höhere Geschwindigkeit beim Scrollen, die dennoch bei weitem nicht an Tempus heranreicht. Auch einige andere häufig benötigte Funktionen sind nicht sehr optimal programmiert: So dauert das Einlesen großer Dateien selbst von der Festplatte auffallend lange. Für ein 100 KByte großes Dokument benötigt Laser C etwa 10 s, während Tempus die Aufgabe in nicht mal 2 s erledigt hat. Auch

das Suchen oder Ersetzen in einer etwas größeren Datei geht leider nicht wie bei Tempus auf Tastendruck. Hier wären Verbesserungen dringend angebracht, damit der Anwender nicht versucht ist, einen externen Editor zu verwenden und damit auf die Vorteile dieses integrierten Systems zumindest teilweise zu verzichten. Das bleibt dem deutschen Benutzer der amerikanischen Programmversion ohnehin nicht erspart, denn der Laser-C-Editor akzeptiert unverständlicherweise im Gegensatz zu seinem Vorgänger keine Umlaute. Bleibt zu hoffen, daß Application Systems als deutscher Distributor dieses Problem lösen wird.

Keine Umweltprobleme

Konnte man beim alten Megamax-System gerade mal Compiler, Linker und ein paar vorgegebene Utilities „lokalisieren“, so läßt sich bei Laser C das gesamte Environment definieren.

Dieses u. a. von Unix bekannte Konzept erlaubt es, Include-Dateien, Libraries und Sources in beliebig definierbaren Pfaden unterzubringen. Man kann sich so z. B. mehrere verschiedene System-Bibliotheken unter verschiedenen Namen halten, ohne irgendwelche Tricks anzuwenden. Genauso lassen sich aber auch häufig benötigte Tools frei in die Menüleiste installieren. Sie können sogar – wie Compiler, Linker und Make – resident im RAM gehalten werden, was bei einigen Programmen jedoch nicht funktionierte, so z. B. bedauerlicherweise bei Tempus.

Einschränkungen entfallen

Was liegt nun näher, als sich schnell aus der Diskettenkiste ein paar alte Megamax-Sources zu kramen, um das neue System auf die Probe zu stellen. Wie zu erwarten, kann man die meisten Sourcecodes direkt neu compilieren, linkern und ausführen. Dennoch gibt es ein paar Änderungen, die man im Hinterkopf behalten sollte: Die Be-


```

Datei example1.c:
    #include "defs.h"
    main()
    {
        int    i;
        for (i=0; i<=HOWOFTEN; i++)
            routine1();
    }

Datei example2.c:
    #include "defs.h"
    main()
    {
        int    i;
        for (i=0; i<=HOWOFTEN; i++)
            routine2();
    }

Datei defs.h:
    #define      HOWOFTEN      10

Datei rout.c:
    routine1()
    {
        printf("Routine1\r\n");
    }

    routine2()
    {
        printf("Routine2\r\n");
    }

Datei makefile:
    #
    # Makefile for EXAMPLE1.PRG and EXAMPLE2.PRG
    #
    Example :: example1.o rout.o
        \megamax\cc.ttp example1.o rout.o -o example1.prg

    Example :: example2.o rout.o
        \megamax\cc.ttp example2.o rout.o -o example2.prg

    example1.o : defs.h
    example2.o : defs.h

```

Bild 2. Beispiel für die Verwendung einer Make-Datei

Ergebnisse der Benchmarks auf einem Atari 1040 ST

	Laser C Version 1.01	Megamax C Version 1.1	Lattice C Version 3.03
Intmath	0.30	0.31	3.09
Realmath (1)	8.47	29.47	11.10
Triglog (1)	8.49	(2)	78.30
Textcorn	83.77	58.44	11.11
Store (3)	4.87	2.30	3.01
Redund1	2.61	5.14	17.41
Redund2	2.31	2.30	11.43
Mult1	1.47	15.10	1.68
Mult2	13.10	15.10	5.44

- (1) Alle Compiler rechneten mit wirklicher Double-Arithmetik; bei Megamax-C mußte zu diesem Zweck die Library DOUBLE.L dazugelinkt werden.
- (2) Bei der Berechnung hängt sich das Programm auf.
- (3) Die Werte wurden mit einer Tandon-Festplatte auf einer leeren Partition ermittelt.

Bild 3. Ergebnisse der Benchmark-Programme aus Bild 4

schränkung eines Code-Segments auf 32 KByte entfällt; damit werden die zahlreichen Overlay-Zeilen (in großen Programmen) überflüssig, sie müssen entfernt werden. Aber nicht nur für das Code-Segment, sondern auch für das Daten- und das BSS-Segment entfällt das 32-KByte-Limit. Sehr erfreulich, da man nun endlich auch große Textmengen und umfangreiche Arrays direkt im Programm verankern kann. Der neue Compiler produziert absoluten Code. Das heißt für den Inline-Assembler, daß Zugriffe auf globale (externe) Variablen nicht mehr wie bisher A4-relativ erfolgen, sondern absolut. Lokale Variablen werden jedoch weiterhin auf einem A6-relativen Stack abgelegt. Ansonsten hat sich am Compiler nur wenig geändert. Recht angenehm ist, daß man jetzt den Datentyp ENUM zur Verfügung hat. Dieser sogenannte Aufzählungstyp ermöglicht Konstruktionen wie:

```

enum { blau, rot, weiß } farbe;

farbe = blau;

```

Inklusive Make

Hat man nun seine Sourcecodes angepaßt, geht's ans Compilieren und Linken. Bei kleineren Programmen geht das noch recht locker mit „Execute Compiler“ und „Execute Linker“ oder ein bißchen komfortabler mit „Run“, das automatisch den Source im aktuellen Window compiliert, zu einem Programm zusammenlinkt und ausführt. Hat man aber ein etwas größeres Projekt mit zahlreichen voneinander abhängigen Quelltexten zu bewältigen, so ist MAKE eine große Arbeitserleichterung. Dieses Konzept läßt sich wohl am besten an einem einfachen Beispiel erläutern (Bild 2). Es existieren zwei Programme (Example1 und Example2), die beide auf ein gemeinsames Includefile (DEFS.H) mit Definitionen und ein Objectfile (ROUT.O) mit gemeinsamen Routinen zugreifen. Wird nun z. B. ROUT.C geändert, müßte man im Handbetrieb zunächst ROUT.C neu compilieren und dann beide Programme neu linken. Durch das aufgelistete Makefile erledigt man das Ganze mit einem simplen Mausklick.

Und ab geht die Post

Was ist nun von den erzeugten Programmen zu halten? Vorweg: Sie übertreffen in puncto Geschwindigkeit die Produkte ihres Vorgängers Megamax 1.0 meist noch etwas (Bild 3). Im Gegensatz zu Megamax-C, das beispielsweise Redundanzen und Multiplikationen mit Vielfachen von 2 (die sich

```

/* Benchmarks for LaserC */
#include "stdio.h"
#include "osbind.h"

long t, t2, *ptr;

main()
{
    printf("Intmeth:      ");
    intmeth();

    printf("Realmeth:     ");
    realmeth();

    printf("Triglog:         ");
    triglog();

    printf("Textscrn:        ");
    textscrn();

    printf("Store:           ");
    store();

    printf("Redund1:         ");
    redund1();

    printf("Redund2:         ");
    redund2();

    printf("Mult1:          ");
    mult1();

    printf("Mult2:          ");
    mult2();
}

gettime()
{
    *ptr = *(long *)0x462;
}

start_timer()
{
    ptr = &t;
    Supexec(gettime);
}

end_timer()
{
    ptr = &t2;
    Supexec(gettime);
    printf("%0.2f seconds\n",
        (double)(t2-t)/(Getraz() == 2 ? 70 : 60));
    Bconln(2);
}

intmeth()
{
    register int x, y;
    register int i = 0;

    x = 0; y = 3;

    start_timer();

    while (i++ < 10000)
        x += ((y * y) - y) / y;

    end_timer();
}

realmeth()
{
    register double x, y;
    register int i = 0;

    x = 0; y = 9.9;

    start_timer();

    while (i++ < 10000)
        x += ((y * y) - y) / y;

    end_timer();
}

triglog()
{
    register double x, y;
    register int i = 0;

    x = 0; y = 9.9;

    start_timer();

    while (i++ < 1000)
        x += sin(atan(cos(log(y))));

    end_timer();
}

```

```

textscrn()
{
    int i = 0;

    start_timer();

    while (i++ < 1000)
        printf("1234567890qwertyuiopkdn", i);

    end_timer();
}

store()
{
    int i = 0;

    FILE *fp, *fopen();

    start_timer();

    if (fp = fopen("D:\Test", "w"))
        while (i++ < 1000)
            fprintf(fp, "1234567890qwertyuiop");

    fclose(fp);

    unlink("D:\Test");

    end_timer();
}

redund1()
{
    register int x, y;
    register long i = 0L;

    y = 1;

    start_timer();

    while (i++ < 200000L)
        x = (y + 0) * (y + 1) + (y * 0);

    end_timer();
}

redund2()
{
    register int x, y;
    register long i = 0L;

    y = 1;

    start_timer();

    while (i++ < 200000L)
        x = y * y + y;

    end_timer();
}

mult1()
{
    register long x;
    register long i = 0L;

    x = 1L;

    start_timer();

    while (i++ < 200000L)
        x = x * 4;

    end_timer();
}

mult2()
{
    register long x;
    register long i = 0L;

    x = 1L;

    start_timer();

    while (i++ < 200000L)
        x = x * 5;

    end_timer();
}

```

Bild 4. Diese Benchmark-Programme führten zu den Ergebnissen in Bild 3

leicht durch Shift-Operationen ersetzen lassen) gar nicht oder wenig optimierte, leistet Laser C hier bessere Arbeit, wie die beiden letzten Benchmarks beweisen (Bild 4). Insbesondere scheinen die Programmierer weiterhin die Fließkomma-Arithmetik gründlich überarbeitet zu haben, während sich beim Schreibzugriff auf Dateien unverändliche Verzögerungen ergeben. Ein anderer Nachteil der fertigen Programme soll auch nicht verschwiegen werden: Sie sind in der Regel doppelt so lang wie die von Megamax 1.0, bleiben damit aber immer noch unter der Länge der von Lattice C produzierten. Ursache dafür ist ganz offensichtlich die Verwendung absoluter Adressen. Man sollte daher – so das Handbuch – die Verwendung globaler Variablen soweit wie möglich vermeiden und stattdessen lokale (Register-) Variablen verwenden, die weiterhin registerrelativ und damit speicherplatzsparend verwaltet werden. Hatte man beim alten Megamax-System nahezu keine Unterstützung beim Debuggen eines Programmes zu erwarten, so ist man

geradezu überrascht von dem Luxus, den Laser C nun bietet: Innerhalb der Shell kann man Programme jederzeit mit Control-Delete aus Endlosschleifen zurückholen. Nach einem solchen Interrupt steht es dem Programmierer frei, entweder in die Shell zurückzukehren, das Programm fortzusetzen oder aber einen Stackdump auszuführen. Letzteres ist eine exzellente Möglichkeit, die Fehlerquelle aufzuspüren, da Laser C die Adresse (bzw. bei aktivierter Linkeroption auch den symbolischen Namen) der laufenden Prozedur und einen Offset zum Anfang der Prozedur anzeigt. Auch Bus-, Adreßfehler und andere Exceptions lassen sich auf ähnliche Weise abfangen: Statt der wenig aussagekräftigen Bomben erscheint eine Dialogbox der Form „68000 Exception 02“ mit der Möglichkeit, zur Shell zurückzukehren, das System neu zu booten oder den beschriebenen Stackdump auszuführen. Diese Option läßt sich selbstverständlich auch ausschalten – wichtig bei der Verwendung eines eigenen Debuggers. Als solcher läßt sich z. B. der

aus dem Digital Research Entwicklungssystem bekannte SID verwenden; wie der Name schon sagt, handelt es sich hierbei um einen symbolischen Debugger. Der Linker des Laser-Systems unterstützt nun genau diese Möglichkeit, indem er optional eine SID-kompatible Symboltabelle in das fertige Programm einfügt. Dadurch wird das Debuggen gerade größerer Programme fast zum Vergnügen.

Insgesamt positiv

Insgesamt hinterläßt Laser C einen positiven Eindruck. Das Handbuch enthält auf nunmehr gut 600 Seiten alle wichtigen Informationen, auch über die verwendeten Dateiformate. Mit der neuen Shell ist es gelungen, Professionalität mit Benutzerfreundlichkeit zu vereinen. Bleibt zu hoffen, daß in der für den Sommer zu erwartenden deutschen Version die kleinen Macken des Systems behoben sind. Dann wird Laser C sicher zu einem neuen Standard für C-Compiler auf dem Atari ST.

Einen 80286-Rechner, der fast so schnell wie ein 386-Computer arbeitet, gibt es vom taiwanesischen Hersteller Mitac. Einen ersten Eindruck von der Arbeitsgeschwindigkeit des MPC 2000VE geben die beiden Bilder mit den Ergebnissen der mc-Testprogramme. Wir vergleichen den Rechner mit dem mc-modular-AT, der mit einer VGA bestückt ist, und mit dem Zenith Z-386. Das Modell 2000VE erreicht die hohe Arbeitsgeschwindigkeit durch einen mit 16 MHz getakteten 80286-Prozessor und einer ausgefeilten Speicher-Architektur, der sogenannten Page/Interleave-Technik. Bei dieser Technik greift die CPU nacheinander auf zwei verschiedene Speicherblöcke zu. Solange sie innerhalb einer Speicherseite zugreift, die beim 2000VE 1 KByte groß ist,

Erster Eindruck: Mitac 2000VE

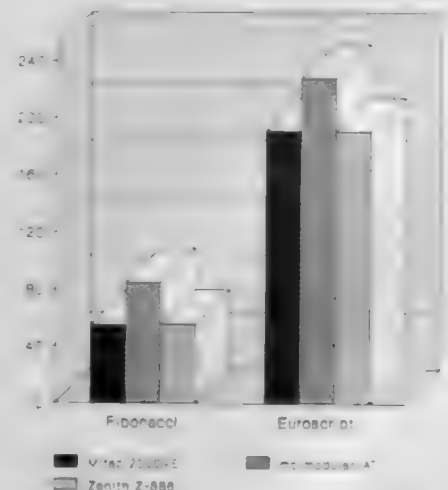
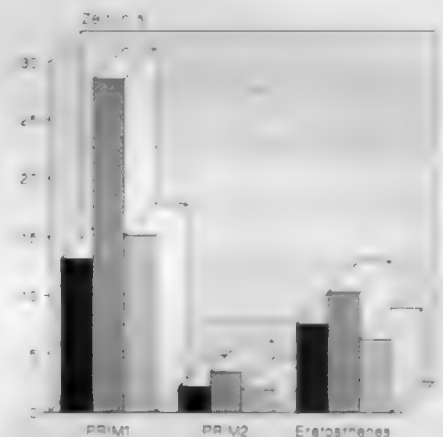
werden keine Wartezyklen eingelegt. Zur Erhöhung der Arbeitsgeschwindigkeit kopiert das System nach einem Kalt- oder Warmstart das BIOS in den Hauptspeicher. Unser Urteil:

Bei einem Preis von um die 6000 DM ist das Modell MPC 2000VE ein schneller, aber dennoch preiswerter Computer. Nachteilig ist, daß im Grundmodell noch keine Festplatte enthalten ist und daher ein Steckplatz durch den Festplatten-Controller verlorengeht.

st

Technische Daten

Typ	Mitac MPC 2000 VE
CPU/Taktfrequenz	80286/16 MHz
Speicher	1 MByte RAM auf Systemplatine (bis zu 8 MByte auf Systemplatine erweiterbar)
Diskettenlaufwerk	1,2 MByte, 5 1/4 Zoll 1,44 MByte, 3 1/2 Zoll
Festplattenlaufwerk	Optional. Ein Steckplatz geht durch den Festplatten-Controller verloren
Steckplätze	5 x 16 Bit, 1 x 8 Bit
Schnittstellen	9poliger Anschluß (COM1 und COM2)
– Seriell	Centronics
– Parallel	MF-II-kompatibel
Tastatur	VGA, EGA, CGA, HGC
Grafikbetriebsarten	377 x 158 x 421 mm ³
Gehäuseabmessungen (B x H x T)	11,5 kg
Gewicht	Schatten-RAM, EMS-Controller
Sonstiges	



Im Vergleich mit anderen Rechnern macht der MPC 2000VE eine gute Figur

Lore Schönrock

Textverarbeitung für den speziellen Einsatz

Manuscript von Lotus

Beim Verfassen umfangreicher Texte werden über die einfachen Textbearbeitungsfunktionen hinaus Textverarbeitungsfunktionen erwartet, die die Seitennumerierung automatisieren, aber auch die Absatznumerierung, Kopfzeilen mit den Kapitelüberschriften mitführen und auch die Überschriften für das Inhaltsverzeichnis übernehmen und das Erstellen von Indizes erleichtern sowie Fußnoten verwalten. Rechtschreibprüfung, Wörterbuch und Trennhilfe sind ebenfalls von großer Wichtigkeit. Zur Visualisierung bestimmter Aussagen im Text werden gerne Tabellen, einfache Grafiken verwandt und sollten darüber hinaus gestaltet werden können.

Das Textprogramm von Lotus bietet dafür ein reiches Leistungsspektrum, das bereits im Hauptmenü angedeutet wird. Neben Bearbeiten und Drucken eines Dokumentes werden hier beispielsweise Wörterbuch und Rechtschreibprüfung aufgeführt. Hinter der Bezeichnung ANSICHT verbirgt sich eine – besonders vorteilhaft für umfangreiches Textmaterial – komfortable Funktion zur Darstellung des Seitenlayouts. Der Bildschirm wird vertikal halbiert, auf der linken Hälfte erscheint die komplette Seite, wie sie im Ausdruck aussehen wird. Der rechte Teil ist horizontal in zwei Bereiche unterteilt. Der obere stellt eine Ansicht auf einen

Im April/Mai brachte die Firma Lotus ihr neuestes Produkt Manuscript auf den Markt. Manuscript ist ein Textverarbeitungsprogramm, konzipiert für spezielle Bedürfnisse, die beim Schreiben von Dokumentationen, Handbüchern, Planungsunterlagen oder Büchern geweckt werden.

bestimmbaren Ausschnitt des daneben stehenden Textes wie durch eine Lupe dar. Darunter wird zur Orientierung die gezeigte Seite des Dokumentes benannt.

Doch bis diese Funktion gebraucht wird, muß erst einmal ein Text geschrieben und bearbeitet worden sein. Dazu tritt das Menü „Dokument bearbeiten“ in Kraft. Der Bildschirm wird – wie in einigen bekannten Textprogrammen auch – für die Texteingabe völlig frei gemacht. Am oberen Bildschirmrand erscheint eine gepunktete Linie, die die Zeilenlänge markiert. Darüber steht der Dateiname. Daneben erfährt man die Gradposition des Cursors, doch leider nicht seine Zeilenposition.

Ab sofort halten die Funktionstasten Manuscript-Befehle abrufbereit. Für den Anfang kann man sich behelfen und sie von der aufzulegenden Schablone ablesen. Alle zehn Funktionstasten sind einfach und in Verbindung mit ALT belegt. Hinter F10 verbirgt sich beispielsweise MENÜ, das in einer Menüleiste am oberen Bildschirm-

rand weitere Funktionen aufzeigt, die über Pfeiltasten oder mit dem jeweiligen Anfangsbuchstaben anzuwählen sind. Global ist beispielsweise eine davon, die nach Anwahl ebenfalls ein Menü aufbaut und unter anderem Optionen ausweist. Will man eine Sicherungskopie automatisch anlegen lassen, muß

man sich bis dahin tasten und die Frage nach Sicherungskopie mit ja beantworten (Bild 1).

F1 ist die Hilfe-Taste und ESC hebt jeden Vorgang wieder auf. Mit ALT-F9 wird gesucht und ersetzt. Für Unterstreichen oder Fettdruck klappt ALT-F6 ein Pull-Down-Menü auf, aus dem das gewünschte Merkmal mit der Cursor-Taste anzuwählen ist. Die Leertaste bestätigt die Wahl. Dieses Prinzip gilt durchgängig überall. Das Programm zeigt eine ausgewogene Kombination zwischen Menü- und Funktionstastensteuerung.

Seine Stärke ist das Format

Besonders anschaulich wird die Auswahl der Befehle bei Manuscript an der Stelle, wo viele andere Programme recht kompliziert, abstrakt und dadurch auch bedienungsunfreundlich werden, nämlich bei der Formatgestaltung. Ich möchte sogar noch einen Schritt weiter gehen und sagen, daß das Formatieren langer Dokumente besonders einfach und aus diesem Grund vorbildlich gelöst ist. Es ist eine Stärke dieses Textprogrammes.

Der Begriff Formatieren beschränkt sich hier nicht nur auf die Gestaltung des rechten Randbereiches. Er bezieht sich auf die Gestaltung des Textes, aber auch auf die Gestaltung der ganzen Seite. Manuscript unterscheidet hierbei zwischen lokalem und globalem Format. Das globale Format beinhaltet alle Parameter wie Tabulatorstopps, Texteinrückungen, Zeilenabstand für das gesamte Dokument. Ob eine Kapitelüberschrift fettgedruckt werden soll oder ob ein anderer Font ausreicht, wird einmal festgeschrieben. Für den gesamten Text

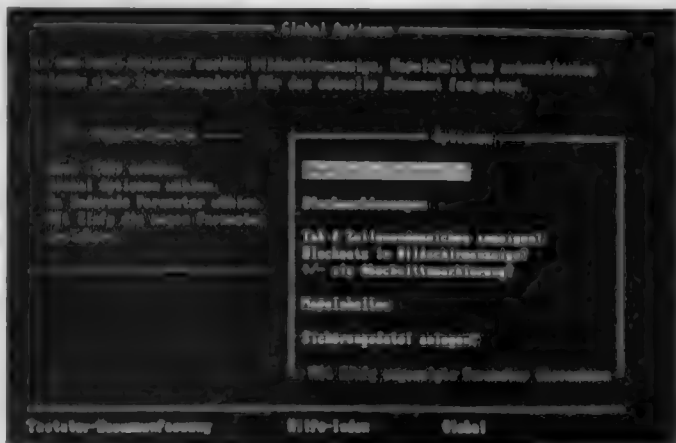


Bild 1. Viele Funktionen bei den Global-Optionen sind über Menüs anzuwählen. Eine nützliche Funktion für lange Texte ist die automatische Anfertigung von Sicherungskopien

Suchen sie einen Partner mit dem Sie wachsen können?

Die INTRA-führender Hersteller von Monitoren - bietet seit Jahren Ihren Kunden eine stetig wachsende Anzahl von Anzeigegeräten und Displays.

Von der Bildröhre bis zum 20" Farb-Videospielmonitor, vom Composite-Monitor bis zum IBM-kompatiblen Monochrom-Monitor und 14" CGA/EGA Multi-SYNC-Bildschirm.

Wir haben nicht nur das Wissen und Können, sondern auch die Qualität.

Wir wollen expandieren und weiterhin qualitativ hochwertige HiTech entwickeln - schon heute fertigen wir bis zu 30.000 Bildschirme monatlich.

Mit 3, 5 Mio Dollar Kapital, 250 Mitarbeitern und einem Werksgelände von knapp 6.000 qm haben wir die besten Voraussetzungen für Wachstum.

Unsere Zukunftspläne erfordern eine verstärkte Präsenz in den Bereichen Ganzseitenmonitor sowie eine Ausweitung der Desk-top-, Modem- und Lap-top-Technologien.

Sie sind in diesem Bereich tätig, verfügen über ausreichende Erfahrung und wollen kooperieren?

Dann sollten Sie mit uns Kontakt aufnehmen. Es wird sich für Sie lohnen.

"VGA" MONOCHROME MONITOR

MODEL 14HP34V

1. 14" FLAT SCREEN
2. PS/2, VGA COMPATIBLE (31.5KHz) ANALOG VIDEO
3. CRT: PAPER-WHITE, AMBER, GREEN

14" MONOCHROME MONITOR

MODEL 14HP33T

1. 14" FLAT SCREEN
2. DUAL FREQ (15.75/18.432KHz)
3. CRT: PAPER-WHITE, AMBER, GREEN
4. WITH REVERSE SWITCH

14" HIGH-RESOLUTION COLOR MONITOR

MODEL

- | | |
|---------|------------------------|
| 14CH113 | (EGA 640 × 350) |
| 14CH114 | (CGA 640 × 200) |
| 14CH115 | (MULTI-SYNC 800 × 600) |
| 14CH116 | (VGA 640 × 350 |
| | 640 × 400 |
| | 640 × 480) |

• "VGA" MONOCHROME MONITOR



• 14" MONOCHROME MONITOR

• 14" HIGH-RESOLUTION COLOR MONITOR



Intra Electronics Co., Ltd.

Room No. 618, 6th Fl., 9, Lane 3,
Min Sheng West Rd., Taipei, Taiwan, R.O.C.

Tel: (02)597-7027 Tlx: 19925 INTRA Fax: 886-2-5418513

IBM, EGA, and CGA are registered trademarks of the International Business Machines Corp.

VED . FCC APPROVED

• OEMs WELCOME !!

TEST

wird bestimmt, wie groß der Zeilenabstand in den einzelnen Absätzen und wie groß er zwischen den Absätzen und zwischen der Überschrift sein soll.

Mit der Schreibmaschine war es problemlos, einen etwas größeren Abstand zwischen Textpassagen zu lassen. Darüber brauchte man keinen Gedanken zu verschwenden. Doch sind die meisten Textprogramme für den Computer gar nicht auf dieses Gestaltungsmoment in der Praxis ausgelegt. Und weil viele Anwender nicht mehr auf den Komfort eines Textprogrammes verzichten können, nehmen sie diesen Umstand hin und machen zwei Zeilenschaltungen nach dem Absatz. Manuscript hat durch seine globale Formatvorgabe das fast Vergessene wieder ausgraben können und dem Anwender bereitgestellt. Am Bildschirm wird die Absatzschaltung durch eine Linie dargestellt.

Die Parameter werden in dafür vorgesehene Bildschirmkarteikarten eingetragen. Da die Seitengestaltung vom Inhalt abhängt, gibt es entsprechend auch verschiedene Karteikarten für die Titelseite, das Inhaltsverzeichnis, für die erste Textseite, die ungeraden und geraden Seiten sowie die Indexseiten. Umfangreiche Texte werden gern wegen der Übersichtlichkeit in einzelne Kapitel und Abschnitte unterteilt, die optisch durch Einrückungen dargestellt werden. Alle Parameter für die Gestaltung der einzelnen Ebenen werden im globalen Format eingetragen. Gibt man die zu nummerierenden Ebenen an, dann übernimmt Manuscript die aufsteigende Nummerierung in der Form 1 für Kapitel, 1.1 für Abschnitt, 1.1.1 für Unterabschnitt.

Der Text läßt sich stufenweise bis auf die Kapitelüberschriften reduzieren. Sicherlich läßt sich die Reihenfolge der Textabschnitte übersichtlicher auf Überschriften-Ebene ändern. Der jeweils dazu gehörende Text wird automatisch mitgenommen. Die graue Minus- und Plus-Taste sowie ALT in Verbindung mit einer Zahl zwischen eins bis neun läßt auf Tastendruck den Text wieder am Bildschirm erscheinen und auch verschwinden. Darüber hinaus lassen sich die Überschriften mit einigen Handgriffen, die zum Aufrufen der Sortierfunktion nötig sind, alphabetisch ordnen. Über ein zweites Fenster kann eine andere Datei an den Bildschirm gerufen und eine Textstelle kopiert werden. Die Fenster-Funktion ist in der Menüleiste von F10 anzuwählen. Mit CTRL-W wechselt man in das andere Fenster. Trotz zahlreicher Änderungen und Verschiebungen des Textes wird Manuscript die Seiten und die Kapitel mit den dazugehörenden Abschnitten entsprechend richtig durchnummerieren.

Bild 2. In Manuscript ist Liniengrafik vorhanden, wie diese Tabelle zeigt

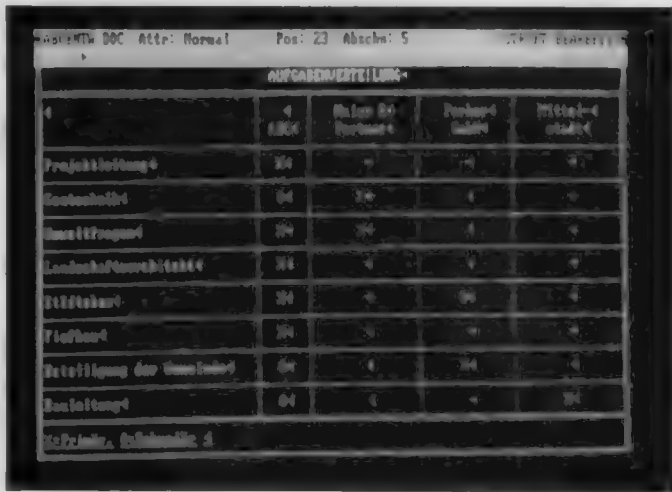
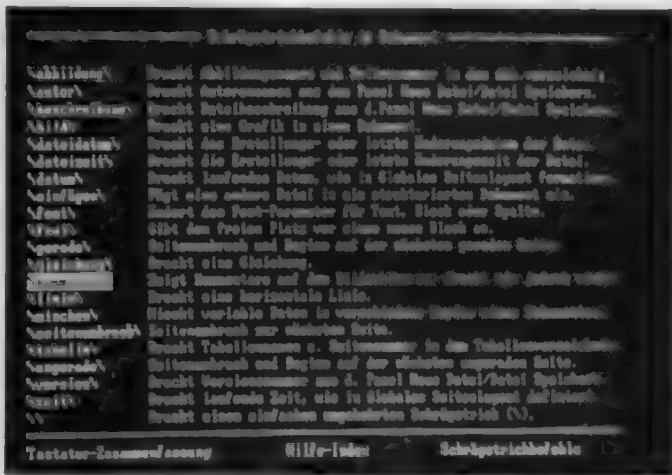


Bild 3. Schrägstrichbefehle werden ins Dokument geschrieben und erst beim Drucken wirksam



Einfache Grafiken wie Organigramme und Tabellen sind ein beliebtes Darstellungsmittel für komplexe Sachverhalte. Rahmen um Textstellen herum können Merk- oder Kernsätze hervorheben. Das sind ebenfalls Gestaltungsmomente, die mit Manuscript genutzt werden (*Bild 2*). Ein wichtiges Hilfsmittel kann es sein, Dokumente aus anderen Programmen und mit fremden Formaten importieren zu können, um sie dann weiterzuverarbeiten. Manuscript importiert Symphony- und 1-2-3- Dateien sowie ThinkTank-, aber auch DataCopy- und MicroTek-Grafikdateien. DCA-Dateien werden im- und exportiert. Die Gestaltung der Kopf- oder Fuß- bzw. Endnoten wird ebenfalls in Karteikarten für das gesamte Dokument vorgegeben. Unabhängig von den Vorgaben für den Text können hier eigene Parameter festgelegt werden. Oftmals soll für die Fußnoten beispielsweise eine kleine Schrift verwendet werden.

Nach Fertigstellung des Textes sollte man in jedem Fall die Rechtschreibprüfung, die das Programm anbietet, nutzen. Erstens werden Rechtschreibfehler behoben, die leicht übersehen werden, wenn man viel an ein und demselben Text arbeitet. Zwei-

tens werden im gleichen Arbeitsgang neue Wörter ins Wörterbuch aufgenommen und somit das häufig gebrauchte Vokabular zur Verfügung gehalten. Erst dann sollte zur Überprüfung des Layouts die Funktion ANSICHT ausgeführt werden.

Die besonderen Parameter

Das lokale Format gewährt partielle Änderungen im Text. Muß eine Passage hervorgehoben sein, dann können dafür im Text selbst Formatparameter, Schriften u. ä. geändert werden. Dafür sind Schrägstrichbe-
 fehle einzusetzen, die das globale Seiten-
 layout außer Kraft setzen. Beispielsweise
 wird der global gültige Font mit

für den folgenden Text bis zum Ende des nächsten Blocks oder zum nächsten Font-Befehl durch den neuen aufgehoben. Verschiedene Fonts haben zur leichteren Auswahl eine Nummer in der „Schriftart-Liste“ bei Manuscript. Die Schrägstrichbefehle ändern für die vorzugebende Textstelle den Font, die Abschnittsnumerierung, den Seitenumbruch, den Seitenumbruch zur geraden oder ungeraden Seite etc. Man benutzt

sie gleichfalls für das Mischen von Daten aus dem Mischdokument, wo die Adressen stehen, in das Textdokument, zum Ändern der Uhrzeit oder des Datums. Erst beim Drucken werden diese Befehle abgearbeitet und veranlassen die gewünschten Änderungen (Bild 3).

Genauso verhalten sie sich auch bei mathematischen Gleichungen, ein Thema, das sehr viele Textprogramme schwer bewältigen. Manuscript bietet zwar eine Lösung mit Hilfe der Schrägstrichbefehle, doch die werden (wie oben bereits erwähnt) erst beim Drucken abgearbeitet. Das Programm erreicht über die Eingabe von Schlüsselwörtern den Ausdruck in wissenschaftlicher Form. Es bedarf jedoch einer intensiven Einarbeitung für diesen Zweck, denn die Syntax des Befehls Gleichung scheint eher kompliziert:

[gleichung „formel“ [breite=breite] [höhe=höhe] [maßstab=maßstab] [typ=text] [abbildung=abbildung]]

Die Verwendung der Elemente muß geübt werden, bevor sich der erste sichtbare Erfolg auf dem Papier einstellt. So verbirgt sich hinter „formel“ eine Reihe von Schlüsselwörtern, deren Zeichenfolge in Anführungszeichen zu setzen ist. abbildung verkörpert den Namen einer Formel oder Gleichung, der in das Tabellenverzeichnis aufgenommen werden soll. breite und höhe stehen für gleichnamige Maße der abbildung. Wird ein Maß davon nicht angegeben, so berechnet es Manuscript proportio-

Nachfolgend einige Beispiele für 'gleichung'-Befehle und deren Ausdruck:

\gleich "a = (-b+-wurzel(b hoch 2 - 4ac)) bruch (2a)" ergibt

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

\gleich "a = (-b+-wurzel(b hoch 2 - 4ac)) bruch (2a)" maßstab=2 ergibt

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

\gleich "im hoch unendlich tief A (t) dt" fährt zu

$$\int_a^{\infty} f(t) dt$$

\gleich "im tief reell nabis 0 (V 0x N) = im tief reell a tief reell"

ergibt

$$\int_0^{\infty} V(V+N) = \int_0^{\infty} a_n$$

\gleich "y tief 2 (a) = sum unter (a=0) über 10 a tief k" ergibt

$$y_k(x) = \sum_{n=0}^{\infty} a_n$$

\gleich "p tief (t) = 12 bruch ((mapel rechen (52 131)) (mapel links (30 143)))" ergibt

$$p_a = r \cdot \frac{12}{52 \cdot 131 - 30 \cdot 143}$$

Bild 4. Formeln und mathematische Gleichungen lassen sich mit Manuscript korrekt ausdrucken. Am Bildschirm sehen sie ganz anders aus

nal zum anderen. typ=text paßt die Größe der Formel der Textzeile an. Da Gleichungen generell in 12 Punkt geschrieben werden, bedeutet das eine Verkleinerung der Formel. maßstab=2 würde die Originalgröße der Gleichung im Ausdruck verdoppeln, wenn höhe und breite nicht vergeben wurden. An dieser Stelle sagt ein Bild wohl mehr als tausend Worte und ein Blick auf Bild 4 strapaziert weniger die Vorstellungskraft.

Alles in allem ist Manuscript ein umfangreiches und leistungsstarkes Produkt, das für den Großeinsatz zum Schreiben langer Texte gedacht ist. Deshalb ist der Leistungsumfang im Formatierungsbereich beachtlich, ebenso die Einfachheit mit der die Funktionen bereitgestellt sind. Das Schreiben von Formeln und Gleichungen ist zwar eine recht abstrakte Arbeit, jedoch hilft die Bedienungsanleitung in diesem Abschnitt

recht ausführlich. Wer seine Materie beherrscht, sollte sich rasch der Arbeitsweise mit den Schrägstrichbefehlen am Bildschirm anpassen können, zumal die Begriffe meist in ihren gängigen Abkürzungen geschrieben werden müssen und auch so dargestellt werden. Im Ausdruck stehen dann stattdessen die Symbole und Zeichen, und zählt nicht schließlich das Resultat? Das Textprogramm kostet 1480 DM und umfaßt acht Disketten, Tastaturschablone, Handbücher und Garantieplan.

Voraussetzungen zur Benutzung von Manuscript von Lotus:

ein IBM-PC, AT oder kompatibler Rechner, mindestens 512 KB, besser 640 KByte RAM (für die Funktion ANSICHT), eine Festplatte und eine Grafikkarte (CGA, EGA, Hercules, etc.), MS-DOS ab Version 2.0

Angenehmes Turbo-Debugging

Speziell für Turbo-Pascal 4.0 kam jetzt sein verbesserter symbolischer Debugger auf den Markt. Über die bisherigen Bearbeitungsmöglichkeiten hinaus besitzt das Produkt nach Angaben des Herstellers Lauer u. Wallwitz, Wiesbaden, eine Multi-Window-Benutzeroberfläche, so daß man nicht ständig zwischen verschiedenen Anzeige-Ebenen hin- und herschalten muß. So werden alle Variablen, der Quell- und der Maschinencode, die Prozedur-Aufrufe und die Prozessor-Register gleichzeitig in verschiedenen Fenstern auf dem Bildschirm dargestellt, wobei der Quellcode entsprechend der Programmausführung mitläuft. Das vollständige Paket und das Update für die große Zahl der Besitzer der Version 2.0 sind wahlweise mit deutscher oder englischer Dokumentation verfügbar.

1: Command

```

>nl
Break: 0000
>nl
Break: 0000
>

```

2: Source

```

6  procedure banana;
7  var
8      i: integer;
9  begin ( banana )
10     x := 3;
11     y := 4;
12     for i := 1 to 3 do

```

3: Calls

```

20  banana;

```

4: Code

```

000F 09E5      MOV BP, SP
0011 800200      MOV AX, 0002
0014 9AAD02794D  CALL 4D79:82AD
0019 83E002      SUB SP, +02
001C C70600000300  MOV WORD PTR [0000], 0003

```

5: Regs

```

AX: 0000
BX: 002C
CX: 0000
DX: 0003
SP: 3FFC
BP: 3FFE
SI: 00DD
DI: 0104
DS: 4E0B
ES: 4E0B
SS: 4E30
CS: 4D6C
IP: 000E
JMP UP EI
PL ZR NA
PE NC

```

Quell- und Maschinencode, alle Variablen, die Prozedur-Aufrufe und die Prozessor-Register erscheinen beim neuen Turbo-Debugger 4.0 gleichzeitig auf dem Bildschirm

Dieter Strauß

Ein schnelles Schneiderlein

Schneiders EGA-AT: Ein modularer Mini-AT

Angenehm klein präsentiert sich der EGA-AT auf dem Schreibtisch (Bild 1). Möglich wird die kompakte Bauweise durch den Einsatz zahlreicher hochintegrierter Bausteine, die den gesamten Computer auf zwei Steckkarten schrumpfen lassen. Eine Steckkarte enthält die CPU und den Arbeitsspeicher von 1 MByte, die andere fungiert als Multi-I/O-Karte. Bestückt ist die CPU-Karte mit einem Chipsatz von Chips & Technologies, der bereits im Rahmen unserer Serie „mc-modular-AT“ [1] beschrieben wurde.

Die Bus-Karte ist wie beim mc-modular-AT rein passiv. Um die Höhe des Gehäuses niedrig zu halten, wurde sie von Schneiders Konstrukteuren in senkrechter Position platziert, so daß sich die Einsteckkarten waagrecht im Gehäuse befinden. Leider gibt es in der Computertechnik fast keinen Vorteil, der nicht einen Nachteil zur Folge hat. Zur Begrenzung der Gehäusehöhe sind nicht mehr als vier Steckplätze vorgesehen, von denen zwei bereits durch die CPU- und die Multi-I/O-Karte belegt sind. Laut Adam Riese bleiben damit dem Anwender nur noch zwei freie Steckplätze für Erweiterungskarten übrig. Zwar hat das System bereits zwei serielle und eine parallele Schnittstelle auf der Multi-I/O-Karte und auch die Video-, die Floppy- und die Festplatten-Schnittstelle sind auf ihr beheimatet, jedoch bei technischen Anwendungen, z. B. bei Meßwertfassungssystemen, die mehrere „exotische“ Schnittstellen benötigen, sind die beiden freien Steckplätze gleich weg. Für Anwendungen im Büro- oder Heimbereich, bei denen lediglich ein kompakter, aber dennoch leistungsfähiger Rechner gewünscht wird, ist der PC2640 jedoch gut geeignet.

80286-CPU unter Dampf

Unsere Benchmark-Programme (Bild 2) belegen, daß er voll im Trend einer Zeit liegt,

Aus gleich vier Computern besteht das neue PC-Programm der Schneider Rundfunkwerke. mc hat sich den EGA-AT genannten PC2640 aus der Produktpalette herausgepickt und genauer angesehen. Mit einer 12,5-MHz-80286-CPU unter der Haube entpuppt er sich als ein schneller modularer AT.



Bild 1. Trotz seiner kleinen Abmessungen ist der PC2640 ein leistungsfähiger Computer der AT-Klasse

In der Anwender hohen Systemdurchsatz fordern, und daß er nicht mit einem langsamen PC der 8088-Generation verwechselt werden darf.

Auch bei der Video-Schnittstelle hat sich einiges getan. Die Standards EGA, CGA und Hercules werden von der Multi-I/O-Karte beherrscht. Auf der Geräterückseite befinden sich zwei DIP-Schalter, die man je nach angeschlossenem Bildschirm – Monochrom- oder EGA-Monitor – einstellen muß. Eine automatische Monitorerkennung wäre hier anwenderfreundlicher, jedoch wegen der Vielzahl am Markt erhältlicher Monitore nicht ganz unproblematisch. Zusätzlich muß noch auf der Multi-I/O-Karte mit dem DIP-Schalter SW1 der jeweilige Monitor und die gewünschte Betriebsart der Video-Schnittstelle eingestellt werden. Das System kann auch automatisch die

Video-Betriebsart (HGC, CGA, EGA) aufgrund des jeweilig ausgeführten Programms erkennen. Gegenüber solchen Automaten bin ich sehr skeptisch eingestellt und bevorzuge daher die manuelle Einstellung.

Bei guten professionellen Programmen kann man häufig während der Installation angeben, in welcher Video-Betriebsart sie arbeiten sollen. Hat man die Multi-I/O-Karte auf EGA eingestellt und einen geeigneten Monitor angeschlossen, muß man bei dem jeweiligen Programm den EGA-Modus auswählen.

War noch bei Schneiders früherer PC-Linie das Netzteil der Systemeinheit im Monitor untergebracht, so daß Monitore anderer Fabrikate nur nach einer umfangreichen Bastelei anschließbar waren, ist beim PC2640 – wie es sich für einen „ordentlichen“ Computer gehört – das Netzteil bereits in der Systemeinheit integriert. Man gewinnt damit die Freiheit, jeden geeigneten am Markt erhältlichen Monitor anschließen zu können. Selbstverständlich gibt es auch von

Schneider geeignete Monitore.

Ein großes Lob verdienen die Handbücher, da sie außer dem Handbuch über die Maus alle in Deutsch verfaßt sind. Das Maus-Handbuch sollte unbedingt auch übersetzt werden.

Während viele PC-Hersteller noch immer ihre Systeme mit MS-DOS 3.2 ausliefern, gibt es für den PC2640 bereits die Version 3.3. Schneider setzt auf den Betriebssystemzusatz GEM und liefert es dementsprechend inklusive der Handbücher zu GEM Write, Paint und Desktop komplett mit. Im Benutzerhandbuch werden anhand zahlreicher Skizzen, die Anschlüsse eines zweiten Disketten-Laufwerks und einer zweiten Festplatte gezeigt.

Für den absoluten Computer-Laien dürfte das insgesamt 106 Seiten starke Handbuch etwas verwirrend sein, da technische De-

TEST

tails, die man zur Inbetriebnahme des Systems gar nicht zu wissen braucht, mit der Anleitung zur Installation und Inbetriebnahme vermischt sind. Den reinen Anwender interessiert z. B. der Aufbau einer Diskette nicht. Er muß nur wissen, wie herum man eine Diskette ins Laufwerk schiebt. Eine kompakte Anweisung zur Inbetriebnahme des Systems am Anfang des Handbuchs mit den entsprechenden Verweisen zu den Seiten, wo die technischen Details genauer erläutert sind, täte dem nicht einschlägig mit Computer-Wissen vorbelasteten Anwender sicher gut.

Es ist allerdings positiv zu bewerten, daß die Bedeutung der zahlreichen Jumper und Anschlüsse der Multi-I/O- und CPU-Karte im Handbuch beschrieben sind. Doch Gu-

Zitat des Monats

„Die Reparatur-Aufträge werden mit den Monteuren gemeinsam ausgedruckt und besprochen.“

Aus einer EDV-Zeitschrift für Handwerker

tes kann man meist noch besser machen. Da die Festplatte des PC2640 bereits formatiert ist und mit MS-DOS versehen geliefert wird, braucht der Anwender nur die Tastatur, die Maus, den Drucker und den Bildschirm anzuschließen und schon kann er das System einschalten.

Die notwendige Einstellung der DIP-Schalter der Video-Schnittstelle nimmt bereits der Händler vor.

Bei der Installation des Systems ist mir aufgefallen, daß die Anschlüsse für den Bildschirm und den Drucker so nahe beieinander liegen, daß man beim Anschließen des Druckerkabels etwas „fummeln“ muß.

Recht ungewöhnlich ist das Anzeigetableau auf der Frontseite des Gehäuses. Sechs Anzeigen teilen dem Anwender jederzeit mit, welche Peripherie des Systems – dazu gehören COM1, COM2, Bildschirm, Drucker, Festplatte und Tastatur – momentan aktiv ist. Eine Betriebsanzeige ergänzt das Tableau. Über den Nutzen dieser „Light-show“ kann man sich streiten, schaden tut sie jedenfalls nicht.

Das Gesamturteil über den PC2640 fällt für Anwendungen im Büro- und Heimbereich, bei denen man mit zwei freien Steckplätzen auskommt, gut aus. Die 3½-Zoll-Disketten-Laufwerke garantieren Zukunftsicherheit, schaffen jedoch Probleme beim Datenaustausch per Diskette mit anderen PC-An-

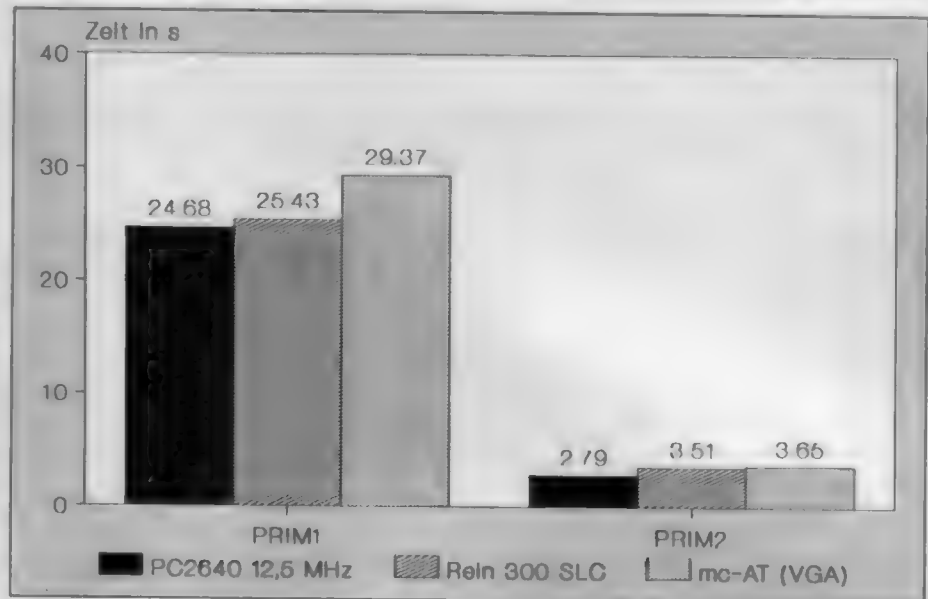


Bild 2. Die hohe Taktfrequenz der CPU macht sich bei den Testprogrammen PRIM1 und PRIM2 bemerkbar

Technische Daten

Typ	PC 2640
Hersteller	Schneider Rundfunkwerke
CPU	80286
Taktfrequenz	6, 10, 12,5 MHz (wählbar)
Coprozessor	Fassung für 80287 vorhanden
Speicher	
– Standard-RAM-Kapazität	640 KByte auf CPU-Karte
– Zugriffszeit	120 ns
Diskettenlaufwerk	
– Standard	1,44 MByte, 3½ Zoll
– Optionen	zweites Laufwerk anschließbar
Festplattenlaufwerk	
– Standard	3½-Zoll-Laufwerk mit 32 MByte
Steckplätze	
– 16 Bit	2 freie Steckplätze
Schnittstellen	
– Maus	wird an COM2 angeschlossen
– Seriell	9poliger Anschluß (COM1 und COM2)
– Parallel	Centronics
Tastatur	MF-II-kompatibel
Bildschirmkarte	auf Multi-I/O-Karte integriert
– Modi	HGC, CGA, EGA
Monitor	Monochrom- oder Farbmonitor
Betriebssystem	MS-DOS 3.3
Reset Schalter	ja
Netzteil-Leistung	145 W
Gehäuseabmessungen (B x T x H)	390 x 420 x 91 mm ³
Gewicht	9,5 kg
Sonstiges	Anzeige-Tafel für diverse Funktionen

wendern und verengen die Softwareauswahl. Da zunehmend mehr Softwarehäuser ihre Programme auch auf 3½-Zoll-Disketten anbieten, wird das letztgenannte Problem immer unwesentlicher. Die Einschränkung des Datenaustausches per Diskette tut schon mehr weh. Statt jedoch an den PC2640 ein externes 5¼-Zoll-Laufwerk anzuschließen, ist es besser, in einen

alten PC mit zwei 5¼-Disketten-Laufwerken, ein 3½-Zoll-Laufwerk einzubauen. Entsprechende Umrüstsätze inklusive Einbaurahmen sind bereits am Markt erhältlich.

Literatur

- [1] Gerd Graf: Der mc-modular-AT. mc 1987, Ausgabe 12, Seite 158.

Gerd Häußler

Ein C-Compiler für die mc-Transputerkarte

Auch heutzutage, im Zeitalter der Megabytes und 32-Bit-Prozessoren, wird noch in vielen Bereichen in Assembler programmiert. Das ist vor allem in Bereich Steuerungstechnik, digitale Signalverarbeitung und Prozeßrechner so, da es sich hier meist um zeitkritische Probleme handelt. Auch die besten optimierenden Compiler können nicht an handoptimierte Assemblerprogramme heranreichen. Aus diesem Grund, und da es sich bei der mc-Transputerkarte in erster Linie um eine professionelle Steuerungskarte für den industriellen Einsatz handelt, gab es in den letzten Ausgaben einen Einstieg in die Transputer-Assemblertechnik. Zur mc-Transputerkarte werden Assembler und Linker kostenlos mitgeliefert.

Mehr als ein EMUF

Sowelt hätte die mc-Transputerkarte nach alter mc-Sitte auch Transputer-EMUF (EMUF= Einplatinencomputer mit universeller Festprogrammierung) heißen können. Sie ist aber mehr als nur ein Einplatinencomputer, da ihre Konzeption und Mächtigkeit auch zuläßt, daß sie als Hauptrechner ausgebaut wird. Die logische Ausweitung von der Assemblersprache aus ist nun der Einsatz der Hochsprache 'C'. C wurde als systemnahe Hochsprache entwickelt, wobei hier immer Schnittstellen zur Assembler-Ebene bereit stehen. Das heißt, es können spezielle Prozeduren durchaus noch in Assembler geschrieben werden, für das 'Drumrum' muß aber nicht auf den Komfort einer Hochsprache verzichtet werden. Natürlich kann ein Programm auch vollständig in 'C' geschrieben werden – hier ist ein Entwickler in der Wahl der Mittel frei. Beim unserem C-Compilersystem handelt es sich um ein Cross Compiler, der auf jedem IBM-PC oder Kompatiblen lauffähig ist. Der Compiler erzeugt Assembler Source Code, der mit dem mitgelieferten Cross Assembler weiterverarbeitet werden kann. Das Standardsystem, das relativ preisgünstig zu erwerben ist, stellt zunächst keine Funktionen zur Parallelverarbeitung bereit.

In den letzten beiden mc-Ausgaben wurde ausführlich auf die Assemblersprache des Transputers eingegangen. Mit diesem Artikel wird ein C-Compiler für die Transputer T-414 und T-800 vorgestellt. Mit ihm können vorhandene Programme leicht auf Transputer portiert werden

Es ist jedoch durchaus möglich, daß sich der versierte Programmierer selbst ein Mini-Betriebssystem schreibt, mit dem es möglich wird, einzelne Prozeduren, die vom Compiler übersetzt wurden, auch parallel auf einem Transputernetzwerk ablaufen zu lassen. Genau diese Möglichkeit, und noch ein paar andere Feinheiten, besitzt das sogenannte Professional System aus unserem Haus, auf das wir im nächsten Heft noch näher eingehen werden.

Der Transputer-C-Compiler für PCs

Der Transputer-C-Compiler enthält bestimmte Optionen, zum Beispiel auch die Option, ob Code für den T-414 oder den T-800 erzeugt werden soll. So wird bei Floatingpoint-Operationen für den T-414 beim Compilieren ein Aufruf für das entsprechende Emulationsprogramm eingesetzt, für den T-800 wird direkt der entsprechende Floatingpoint-Befehl erzeugt. Die „Integerleistung“ dieser beiden Prozessoren ist identisch. Bei Floatingpoint-Berechnungen kommt der T-800 mit seinem integrierten Floating-Point-Prozessor allerdings ganz groß heraus. Damit es nicht bei leeren Versprechungen bleibt, seien hier die Ergebnisse einiger Benchmarks mitgeteilt, die jeder objektiv nachprüfen kann. Als erstes Beispiel für den Test des Compilers wurde der das Programm 'Sieve' (Sieb des Eratosthenes) nach Byte ausgewählt. Bild 1 zeigt das C-Programm. Bild 2 das, was der Compiler nach dem Aufruf

```
cct -x2 sieve
```

daraus macht. Die Option -x erlaubt es, die Zeilen des zu übersetzenden Programms als Kommentare in die Assemblerquelle zu

übernehmen. Wird dabei der Schalter auf '2' gestellt, so wird zusätzlich noch vor jeder Funktion das Stack-Layout für die lokalen Variablen mit ausgegeben – d. h., es wird die Zuordnung zwischen Variablenname und Offset der Variablen im Workspace angegeben. So läßt sich relativ leicht verfolgen, was der

Compiler aus dem Quellcode macht. Nachdem dann die Kommandos

```
asst sieve (assemblieren) und
lnkt c0 sieve libc -osieve (linken) und
optim sieve (optimieren)
```

eingegeben wurden (asst, lnkt und optim sind bei der Lieferung der mc-Transputerplatine dabei; ein solcher Lauf wird natürlich am besten unter einer geeigneten Batchprozedur absolviert), steht die ausführbare Datei 'sieve.oc' im aktuellen Directory und kann von der mc-Transputerkarte geladen werden. Die Spannung war natürlich sehr groß, als wir das Programm zum ersten Mal auf ablaufen ließen – als Vergleich dazu haben wir die Messergebnisse des Benchmarks auf dem Atari ST mit dem Mark Williams C-Compiler, und auf einem AT-286 mit 13 MHz, also einer recht flotte Kiste, unter Turbo-C ermittelt. Bild 3 zeigt die Messergebnisse, wobei noch zu unterstreichen ist, daß es sich beim Transputer um einen 20MHz-Typ handelte. Es war eigentlich klar, daß der Sieger von vornherein schon feststand, aber daß noch ein voller Faktor 2 zum 13-MHz-286 herauskam, war doch einigermaßen verblüffend. Dies Ergebnis bedeutet, daß eine RISC-Maschine mit 20 MHz die Performance einer CISC-Maschine mit 26 MHz erreicht. Dabei muß natürlich noch sehr eindringlich darauf hingewiesen werden, daß ein Schaltungsaufbau eines 20-MHz-Transputers ungleich einfacher vonstattengeht, als zum Beispiel eine entsprechende 20-MHz-Beschaltung eines 80286-Systems.

Occam gegen Transputer-C

Als nächsten Vergleich haben wir uns ein Rennen Occam gegen C ausgedacht. Hier

```

/* Eratosthenes Sieve Prime Number Program in C from Byte January 1983 */
#define true 1
#define false 0
#define size 8191
int flags[size];

int main()
{
    int i, prime, k, count, iter;

    printf("100 iterations\n");
    for(iter = 1; iter <= 100; iter++)
    {
        count = 0;
        for(i = 0; i <= size; i++) /* prime counter */
            flags[i] = true; /* set all flags true */
        for(i = 0; i <= size; i++)
        {
            if(flags[i])
            {
                prime = i + i + 3; /* twice index + 3 */
                for(k = i + prime; k <= size; k += prime)
                    flags[k] = false; /* kill all multiples */
                count++; /* primes found */
            }
        }
    }
    printf("sieve ends.");
}

```

Bild 1. Das Sieb des Eratosthenes in C_1 so wie wir es in der Zeitschrift Byte gefunden haben

```

; C-Crosscompiler   cct fuer Transputer T-414/T-800
;
; (c) 1988 by CESYS GmbH
; Henkestrasse 8
; 8520 Erlangen
;
; Tel. 09131/21098
;
; extern code      _fpuitor14, _fpmul1414, _fpstn1414,
;                  _fpdiv1414, _fpnegate1414, _fpeg1414,
;                  _fsub1414, _fpldn1414, _fpgr1414
;                  _fprev1414, _fpftoi1414, _fpst1414
;

```

Bild 2. Der Cross Compiler macht aus einem C-Programm ein Transputer-Assembler-Programm

August 1988

TEST

Atari-ST	At-286	mc-Transputer
55	11 s	5,5 s

Atari-ST mit Mark Williams C-Compiler
At-286 mit Turbo-C
mc-Transputer mit Csys C-Compiler

```

VAL F.SIZE IS 8191:
[10000]BYTE wastel:
[F.SIZE+1]BOOL flags:
[10000]BYTE waste2:
INT prime, count, lauf:

SEQ
  init.io()
  text.out(14,"100 iterations")
  crlf.out()

  SEQ iter = 0 FOR 100
    SEQ
      count := 0
      SEQ a = 0 FOR F.SIZE
        flags[a] := TRUE
      SEQ a = 0 FOR F.SIZE
        SEQ
          IF
            (flags[a])
              SEQ
                prime := ((a + a) + 3)
                lauf := a + prime
                WHILE (lauf <= F.SIZE)
                  SEQ
                    flags[lauf] := FALSE
                    lauf := lauf + prime
                count := count + 1
                (NOT flags[a])
                SKIP
          char.out('I')
          text.out(11,"Sieve ends.")

```

Atari-ST	At-286	mc-Transputer
1000	2500	4300

Atari-ST mit Mark Williams C-Compiler
At-286 mit Turbo-C
mc-Transputer mit Csys C-Compiler

Bild 3. Das sind die Meßergebnisse von Sieve mit einem Atari, einem 286er (13 MHz) und der mc-Transputerkarte

Bild 4. Sieve, in Occam übersetzt

Bild 5. Der Dhrystone-Mix liefert die Anzahl der durchgeführten Operationen pro Zeiteinheit

puter-Karten die bisherige Steuerung ablösen sollen. Wenn auch nach einer solchen Umsetzung sicher noch viel Anpassungsarbeit in Bezug auf die konkreten System-Ressourcen (IO-Adressen und vieles mehr...) zu leisten ist, dürfte die Entscheidung für den Einsatz der mc-Transputerkarten wegen der Portierbarkeit sicher leichter fallen, als wenn man mit der Umstellung auch völlige Neuprogrammierung in Kauf nehmen müßte.

Der Dhrystone-Mix

Bild 5 zeigt das Testergebnis zum sogenannten Dhrystone-Mix. Dieser Test ist umfangreicher als das 'Sieve', denn dabei werden alle Funktionen geprüft, die nach allen Erfahrungen im Rahmen eines heutigen 'durchschnittlichen' Programmes (für klassische CPUs) in Anspruch genommen werden. Dieser Test hat mehr Aussagekraft als 'Sieve'. Wieder treten dieselben Rivalen wie bei 'Sieve' zur Konkurrenz an. Zum Schluß nun noch ein paar Bemerkungen zum Compiler an sich. Wie schon gesagt, handelt es sich um einen Cross Compiler, der auf IBM-PCs abläuft. Als Bonbon wird demnächst noch ein Sourcecode Debugger für den Compiler vorgestellt werden. Das Thema Debugger wird bei vielen angebotenen Compilern noch recht stiefmütterlich behandelt, wobei natürlich das eigentliche Debugging eines Programms immer noch im Kopf des Programmierers bei der Lektüre des Programmlistings ablaufen sollte. Bei einigen kniffligen Problemem ist ein Sourcecode-Debugger jedoch sehr nützlich, und kann so manche Mannwochen an Entwicklungszeit einsparen helfen. Der Compiler ist ein reiner Einpass-Compiler und arbeitet relativ schnell. Da der Compiler vollständig in C geschrieben wurde, soll er sich demnächst selbst auf den Transputer 'crosscompilieren' – dann dürfte sich seine Geschwindigkeit wohl nochmals erhöhen. Da der Compiler, wie vorhin schon erwähnt, Assembler-Texte erzeugt, die auf Wunsch auch die C-Texte als Kommentare enthalten, ist er sehr Systemprogrammiererfreundlich. In Verbindung mit dem Assembler Asst steht ein Entwicklungssystem zur Verfügung, das Einsteigern einen maschinennahen Weg in das Reich der Transputer bahnt. Dies ist nicht unbedingt gegen das etwas umständliche Occam-System TDS von INMOS gerichtet, das dennoch bei uns ab jetzt wohl in der Schublade verschwinden dürfte.

Die Beschreibung des „Professional System“ finden Sie in der nächsten Ausgabe von mc. Sie erscheint am 29. 8. 1988

könnte die Kritik aufkeimen, daß dabei Äpfel mit Birnen verglichen werden. Die Sprache Occam bietet jedoch für die Testaufgabe ohne Parallelisierung praktisch dieselben Sprachelemente wie C, womit uns dieser Vergleich noch einigermaßen gerecht zu sein scheint. Die Occam-Version von SIEVE ist in Bild 4 gezeigt. Es ist deutlich zu sehen, daß die Sprachen C und Occam in diesem Beispiel zu vergleichen sind. Da jedoch Occam im Gegensatz zu C keine Strukturen und beliebig verknüpfte Datentypen enthält, können nicht alle C-Programme so leicht in Occam umgewandelt werden. Dies gilt insbesondere für den später noch erwähnten Dhrystone Benchmark. Das Occam-Sieve lief ungefähr um zehn Prozent langsamer ab, als die C Version. Dies liegt allerdings wahrscheinlich nicht daran, daß der Occam-Compiler schlechteren Code als der C-Compiler erzeugt, sondern daß in Occam standardmäßig eine Laufzeitüberprüfung der Array-

grenzen vorgenommen wird. Eine solche Überprüfung kann in C aufgrund der Sprachdefinition nicht vorgenommen werden.

Mit C kompatibel

Für unser C-System kann als unumstrittener Vorteil gegenüber Occam die Kompatibilität mit C-Versionen für andere CPUs ins Feld geführt werden. Hat man einmal C auf einer UNIX-Anlage gelernt, so kann man auch den Transputer in C programmieren. Noch wichtiger ist allerdings die Tatsache, das bestehende Programme ohne Aufwand auf die neue Hardware portiert werden können. Man denke hier zum Beispiel an eine Steuerung, die in weiser Voraussicht in C programmiert wurde. In diesem Fall muß das vorhandene Programm einfach nur neu compiliert, assembliert und gelinkt werden, wenn aus Gründen der Steigerung der Performance eine oder mehrere Trans-

Stefan Demmig

Schnelle serielle Schnittstelle

Zur Programmierung der seriellen Schnittstelle auf Betriebssystemebene stellt MS-DOS das Mode-Kommando zur Verfügung. Leider können damit nur Baudraten zwischen 110 und 9600 Baud und Wortlängen von 7 oder 8 Datenbits ausgewählt werden. Die Hardware eines IBM-kompatiblen PCs ist aber in der Lage, Übertragungsraten bis 115 kBaud und 5...8 Datenbits zu verarbeiten. Das Programm SETCOM (Bild 1) leistet die erforderlichen Einstellungen. Es wurde auf dem XT-kompatiblen Rechner Toshiba T1200 entwickelt und sollte auf allen IBM-kompatiblen PCs laufen. Es ist als funktionsfähiges Grundmodul für eigene Weiterentwicklungen zu verstehen.

Funktionsprinzip der seriellen Schnittstelle

Die Parameter der seriellen Schnittstelle sind durch die Programmierung zweier Register des Bausteins 8250 einstellbar. Der 16-Bit-Teilerspeicher dient zur Baudraten-Erzeugung und das Leitungssteuerregister zur Einstellung der sonstigen Parameter. Die Port-Adressen der beiden seriellen Schnittstellen liegen üblicherweise bei 3F8H bis 3FFH für COM1 und 2F8H bis 2FFH für COM2. Die Basisadresse für die erste serielle Schnittstelle (COM1) wird im BIOS im Speicherwort 0:400H abgelegt, die Basisadresse der zweiten seriellen Schnittstelle (COM2) im Speicherwort 0:402H. Um das Programm SETCOM auf die zweite serielle Schnittstelle anzuwenden, muß man lediglich im Definitionsteil die Adresse 400H auf 402H zu ändern. Der Teilerspeicher (Port-Adressen 3F8H = Low-Byte und 3F9H = High-Byte) kann mit Werten zwischen 1 und 65535 (0FFFFH) belegt werden und erzeugt dann aus der Quarz-Frequenz (115200 Hz) die Baudrate nach der Formel:

$$\text{Baudrate} = (\text{Quarz-Frequenz}) / \text{Teiler}$$

Dadurch sind Baudraten von 1.76 bis 115200 Baud einstellbar. In der Tabelle sind die Teiler-Werte für die „klassischen“ Baudraten bzw. die möglichen Baudraten über 9600 Baud aufgeführt. Damit ergeben sich im Bereich kleiner Baudraten (30 bis 100 Baud) interessante Möglichkeit zur

Die größte Übertragungsgeschwindigkeit der standardmäßigen seriellen Schnittstelle eines PCs beträgt 9600 Baud. So steht es in den Handbüchern des MS-DOS-Betriebssystems. Mit einem kleinen Programm läßt sich die Übertragungsgeschwindigkeit bis auf 115200 Baud steigern.

Feinjustierung. Das dürfte für Funkamateure nützlich sein, die Funksignale von mechanischen, gelegentlich schlecht justierten Fernschreibern empfangen wollen.

Das Leitungssteuerregister (Port-Adresse 3FBH) enthält ein Umschaltbit für den wechselweisen Zugriff auf die Datenre-

```

PAGE 60,132
TITLE SETCOM
SUBTTL Programm zum Einstellen von COM1:

;Programm SETCOM
;Dient zum Programmieren der seriellen Schnittstelle COM1:

;Aufruf:
;SETCOM <CE> gibt den Status von COM1: aus
;SETCOM <Baudrate>,<Parität>,<Datenbits>,<Stopbits> programmiert Schnittstelle

;Baudrate: 20..115200
;Parität: N(o), E(ven), O(dd)
;Datenbits: 5..8
;Stopbits: 1,2 (bei 5 Datenbits bedeutet 2 1.5!)

;Jeder Parameter kann fehlen.
;Die Kommas sind unbedingt notwendig, wenn weitere Parameter folgen.
;2.B. SETCOM ,,6 setzt 6 Datenbits, alles andere bleibt unverändert.

;Bei Eingabefehlern wird der DOS-Fehlercode 2 übergeben.

-----

;Definitionen
; Typ-Definitionen
LineCtrl RECORD Gen:1,Break:1,Stick:1,Even:1,Enable:1,Stop:1,Data:2
; Adressen:
Com1Adr = 400h ;Portadresse für COM1: bei 0:400h
LineCtrlReg = 3 ;als Offset von Com1Adr
CmdLine = 80h
; Konstanten:
CrystHi = 1 ;Quarz-Normal =
CrystLo = 0C200h ;115200 Hz (1C200h)

-----

;Code-Segment

SetCom SEGMENT
ASSUME cs:SetCom,ds:SetCom,es:SetCom,ss:SetCom
ORG 100h

; Hauptprogramm -----

SetComEntry: ;Initialisierung
MOV sp,OFFSET Stack ;Stackpointer initialisieren
MOV bx,(Stack-SetComEntry)/10h+11h ;Benötigten Speicherplatz reservieren
MOV ah,4Ah
INT 21h
JNC MemoryOk ;Sprung, wenn Platz verfügbar
MOV ax,4C01h ;Abbruch mit Speicherfehler (1)
INT 21h
MemoryOk:
MOV dx,0 ;BIOS-RAM-Segment = 0
MOV es,dx ;es zeigt aufs BIOS-RAM-Segment
MOV si,CmdLine+1 ;LinePointer auf erstes Zeichen

```

Bild 1. Ein kleines Assembler-Programm zum Tuning der seriellen Schnittstelle

```

CLD
CALL NextChar
CMP al,0Dh
JNE BaudrateIn
JMP DisStatus

BaudrateIn:
CMP al,'.'
JE ParityIn
DEC al
CALL NumberIn
CALL DelimCheck
CMP dx,0
JE BaudrateIn_Branch1
CMP dx,1
JE BaudrateIn_Branch4
JMP Error
BaudrateIn_Branch4:
CMP ax,CrystLo
JNE BaudrateIn_Branch2
JMP Error
BaudrateIn_Branch2:
MOV ax,1
JMP SHORT BaudrateIn_Branch3
BaudrateIn_Branch3:
CMP ax,20
JAE BaudrateIn_Branch5
JMP Error
BaudrateIn_Branch5:
MOV cx,ax
MOV ax,CrystLo
MOV dx,CrystHi
DIV cx
BaudrateIn_Branch3:
MOV cx,ax
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
IN al,dx
OUT dx,al
MOV dx,[es:Com1Adr]
MOV ax,cx
OUT dx,ax
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
IN al,dx
AND al,NOT MASK Gen
OUT dx,al

ParityIn:
CALL NextChar
CMP al,0Dh
JNE ParityIn_Branch1
JMP DisStatus
ParityIn_Branch1:
CMP al,'.'
JE DatabitsIn

:Scan-Richtung vorwärts
:erstes Zeichen lesen
:CR entspricht Eingabe-Ende
:kein Eingabe-Ende
:Eingabe-Ende

:Auswertung Eingabe für Baudrate
:Komma bedeutet keine Daten
:keine Baudrate
:LinePointer zurück auf erste Ziffer
:Zahl für Baudrate lesen
:Prüfung korrekte Begrenzung
:Prüfung höherwertiges Wort der Zahl
:Baudrate < 65536

:Baudrate > 65536, weiter prüfen
:Baudrate viel zu groß
:Baudrate > 65536 (dx = 1)
:Prüfung niederwertiges Wort
:115200 >= Baudrate >= 65536
:Baudrate > 115200
:Baudrate muß 115200 sein
:Baudrate = 115200
:Baudrate setzen
:Baudrate < 65536
:Baudrate soll >= 20 sein
:Baudrate > 20
:Baudrate zu klein
:65536 > Baudrate >= 20
:cx = Baudrate
:dx:ax = Quarz-Frequenz
:ax = dx:ax / cx (Generator-Wert)
:BaudrateGen setzen
:Adresse für LineCtrlReg laden

:Zugriff BaudrateGen EIN
:BaudrateGen schreiben

:Adresse für LineCtrlReg laden

:Zugriff BaudrateGen AUS
OUT dx,al

:Auswertung Eingabe für Parität
:nächstes Zeichen lesen
:CR entspricht Eingabe-Ende
:kein Eingabe-Ende
:Ende Eingabe
:Komma bedeutet keine Daten
:keine Paritätsdaten

CALL DelimCheck
CMP al,'.'
SUB al,20h
ParityIn_Branch2:
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
CMP al,'N'
JNE ParityIn_Branch3
IN al,dx
AND al,NOT MASK Enable
OUT dx,al
JMP SHORT DatabitsIn
ParityIn_Branch3:
CMP al,'g'
JNE ParityIn_Branch4
IN al,dx
OR al,MASK Enable + MASK Even
OUT dx,al
JMP SHORT DatabitsIn
ParityIn_Branch4:
CMP al,'0'
JNE Error
IN al,dx
OR al,MASK Enable
AND al,NOT MASK Even
OUT dx,al

DatabitsIn:
CALL NextChar
CMP al,0Dh
JE DisStatus
CMP al,'.'
JE StopbitsIn
CALL DelimCheck
SUB al,'5'
JS Error
CMP al,3
JA Error
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
MOV cl,al
IN al,dx
AND al,NOT MASK Data
OR al,cl
OUT dx,al

StopbitsIn:
CALL NextChar
CMP al,0Dh
JE DisStatus
CALL DelimCheck
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
CMP al,'1'
JNE StopbitsIn_Branch2
IN al,dx

```



```

AND al,NOT MASK Stop
OUT dx,al
JMP SHORT DispStatus
StopbitsIn_Branch2:
CMP al,'2'
JNE Error
IN al,dx
OR al,MASK Stop
OUT dx,al
JMP SHORT DispStatus

Error:
MOV bx,2
MOV dx,OFFSET ErrMsg
MOV cx,16
MOV ah,40h
INT 21h
MOV ax,4C02h
JMP SHORT DispStatus1

DispStatus:
MOV ax,4C00h
DispStatus1:
PUSH ax
MOV dx,OFFSET Com1
MOV ah,9h
INT 21h

BaudrateOut:
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
IN al,dx
PUSH ax
OR al,MASK Gen
OUT dx,al
MOV dx,[es:Com1Adr]
IN ax,dx
MOV cx,ax
MOV dx,[es:Com1Adr]
ADD dx,LineCtrlReg
POP ax
PUSH ax
OUT dx,al
CMP cx,1
JNE BaudrateOut_Branch1
MOV dx,OFFSET HiSpeedStr
MOV ah,9h
INT 21h
JMP SHORT BaudrateOut_Branch2
BaudrateOut_Branch1:
MOV ax,CrystLo
MOV dx,CrystHi
DIV cx
CALL NumberOut
BaudrateOut_Branch2:
MOV dl,'.'

;Löschen Stopbit-Flag: 1 Stopbit
;fertig, Statusausgabe
;Prüfung 2 und Fehler
;Keine 2, Fehler
;Setzen Stopbit-Flag: 2 Stopbits
;fertig, Statusausgabe
;Fehlerbehandlung
;Auswahl Fehlergerät
;Ausgabe Fehlermeldung
;Länge der Fehlermeldung
;Daten für Ende mit Eingabefehler (2)

;Ausgabe Status
;Daten für Ende ohne Fehler (0)
;Einsprung für Fehlerroutine
;Retten Daten für Ende
;Ausgabe 'COM1:'

;Ausgabe Baudrate
;Adresse für LineCtrlReg laden
;Lesen LineCtrlReg
;Merken Inhalt LineCtrlReg
;Freigabe BaudrateGen

;Adresse für BaudrateGen laden
;Lesen BaudrateGen
;Ablegen in cx
;Adresse für LineCtrlReg laden
;Erinnern Inhalt LineCtrlReg
;Erneutes Merken Inhalt LineCtrlReg
;Rücksetzen LineCtrlReg
;wenn cx = 1, dann Baudrate = 115200
;Baudrate < 65536
;Ausgabe Baudrate=115200 (Konstante)

;Komma ausgeben und weiter
;Baudrate < 65536
;Laden dx:ax mit Quarz-Frequenz
;Teilen durch cx, gibt Baudrate
;Ausgeben Baudrate
;Komma ausgeben

;Unterprogramme -----
;
; NextChar PROC
; Funktion: Sucht nächstes Zeichen außer SPACE und Ctrl-I

```

```

MOV ah,2h
INT 21h

;Ausgabe Parität
;Erinnern Inhalt LineCtrlReg
;kopieren LineCtrlReg von al in ah
;Erneut merken
;Maskieren Parity Enable Bit
;Parität gesetzt
;Parität nicht gesetzt, also 'g'
;Ausgabe Parität
;Parität gesetzt, prüfen Even/Odd
;Maskieren Even Parity Bit
;Even Parity gesetzt
;Even Parity nicht gesetzt, also '0'
;Ausgabe Parität
;Even Parity gesetzt
;'g' ausgeben
;Ausgabe Parität

;Ausgabe Komma

;Ausgabe Datenbits
;Erinnern LineCtrlReg
;wieder merken
;Maskieren Datenbits
;0='5'..3='8'

;Ausgabe Datenbits

;Ausgabe Komma

;Ausgabe Stopbits
;Erinnern LineCtrlReg
;Maskieren Stopbit-Flag
;gesetzt, also 2 Stopbits
;1 Stopbit
;Ausgabe + Ende
;2 Stopbits

;Ende der Ausgabe
;Ausgabe Stopbits

;Ausgabe CR+LF (Zeilenende)

;Holen Daten für Ende

; Unterprogramme -----
;
; NextChar PROC
; Funktion: Sucht nächstes Zeichen außer SPACE und Ctrl-I

```

```

; Eingabe: si=LinePointer
; Verändert: Flags
; Ausgabe: si=LinePointer, al=Zeichen
NextChar_Loop1:
    LODSB
    CMP al, ' '
    JE NextChar_Loop1
    CMP al, 9h
    JE NextChar_Loop1
    RET
NextChar ENDP

; nächstes Zeichen mit LinePointer lesen
; Prüfen SPACE
; nächstes Zeichen
; Prüfen auf Ctrl-I
; nächstes Zeichen

DelimCheck PROC
; Funktion: Prüft nächstes Zeichen als Begrenzer: SPACE, Ctrl-I und '*'
; Eingabe: si=LinePointer
; Verwendung: Flags
; Ausgabe: si=LinePointer, al=Zeichen
    CMP [si], BYTE PTR '*'
    JE DelimCheck_Branch1
    CMP [si], BYTE PTR 9
    JE DelimCheck_Branch1
    CMP [si], BYTE PTR ' '
    JE DelimCheck_Branch1
    CMP [si], BYTE PTR 0Dh
    JE DelimCheck_Branch2
    JMP Error
DelimCheck_Branch1:
    INC si
DelimCheck_Branch2:
    RET
DelimCheck ENDP

NumberIn PROC
; Funktion: liest Ziffernfolge als 16-Bit-Integer
; Eingabe: si=LinePointer
; Verwendung: Flags, bx, cx, dx
; Ausgabe: si=LinePointer, ax=Integer
    MOV dx, 0
    MOV ax, dx
    MOV bx, 10
    PUSH ax
NumberIn_Loop1:
    LODSB
    CMP al, '0'
    JB NumberIn_Branch1
    CMP al, '9'
    JA NumberIn_Branch1
    SUB al, '0'
    MOV cl, al
    MOV ch, 0
    POP ax
    MUL bx
    ADD ax, cx
    ADC dx, 0
    CMP dx, 0
    JNZ NumberIn_Branch2
    JMP SHORT NumberIn_Loop1
NumberIn_Branch1:
    ; Zeichen ist nicht Ziffer
    ; LinePointer zurück auf Begrenzer
    ; Integer zurückholen
    ; Wert ist zu groß (Abbruch)
    RET
NumberIn ENDP

NumberOut PROC
; Funktion: Gibt 16-Bit-Integer auf den Bildschirm aus
; Eingabe: ax=Integer
; Verwendung: Flags, bx, cx, dx
; Ausgabe: keine
    MOV bx, 10000
    MOV cl, 0
NumberOut_Loop1:
    MOV dx, 0
    DIV bx
    PUSH dx
    CMP al, cl
    JE NumberOut_Branch1
    MOV cl, 0F7h
    MOV dl, '0'
    ADD dl, al
    MOV ah, 2
    INT 21h
    MOV ax, bx
    MOV dx, 0
    MOV bx, 10
    DIV bx
    MOV bx, ax
    POP ax
    CMP bx, 0
    JNE NumberOut_Loop1
    RET
NumberOut ENDP

; Texte und Daten -----
ErrMsg:
    DB 'Eingabefehler', 7
CrLf:
    DB 0Dh, 0Ah, '$'
Com1:
    DB 'COM1: ', '$'
HiSpeedStr:
    DB '115200', '$'
    DB 64 DUP(?)
Stack:
SetCom ENDS
END SetComEntry

```

gister der Schnittstelle und den 16-Bit-Teilerspeicher, die auf den selben Port-Adressen „übereinandergelegt“ sind. Im übrigen wird durch das Leitungssteuerregister die Datenwort-Länge, die Parität und die Anzahl der Stoppbits bestimmt. Die Bedeutung der einzelnen Bits und die möglichen Einstellungen sind in *Bild 2* zu sehen. Da das Datenwort 5 bis 8 Bit lang sein darf, läßt sich damit der 5-Bit-Baudot-Code der alten Postfernschreiber verarbeiten. Deshalb werden auch nur 1,5 Stoppbits gesendet, wenn 5 Datenbits und 2 Stoppbits programmiert worden sind.

Beide Register sind schreib- und lesbar, wodurch sich die Programmierung vereinfacht, da man keine zusätzlichen Speicherstellen für die Ablage der Einstellungsdaten verwenden muß.

Mit dem Programm SETCOM kann man den Zustand der seriellen Schnittstelle abfragen. Dazu startet man es ohne Angabe von Parametern mit SETCOM.

Durch Anhängen von Parametern an den Programm-Namen in der Kommandozeile können einige oder alle Parameter der Schnittstelle eingestellt werden. Die eingegebenen Parameter müssen durch Kommas, Leerzeichen oder Tabulator-Zeichen (Ctrl-I) voneinander getrennt werden. Dabei werden die einem Komma folgenden Leer- und Tabulator-Zeichen überlesen.

Die Parameter-Folge ist folgendermaßen festgelegt: Baudrate (20..115200 Baud), Parität (N, E, O), Zahl der Datenbits (5..8), Zahl der Stoppbits (1, 2). Will man beispielsweise nur die Zahl der Datenbits einstellen, muß man vorausgehende Parameter-Plätze mit Kommas abtrennen; nachfolgende Parameter müssen nicht angegeben werden:

SETCOM „6

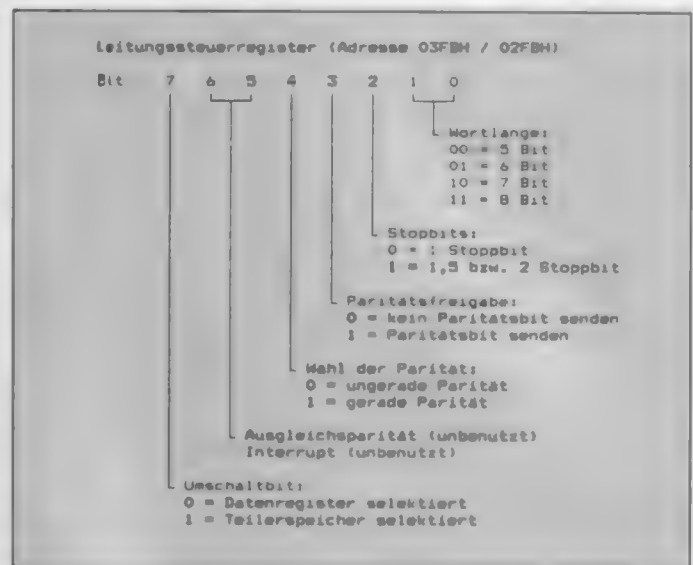
Alle unbesetzten Parameter werden nicht verändert.

Die Eingabe und Darstellung der Daten ist auf ganze Zahlen beschränkt. Bei 5 Datenbits werden 2 Stoppbits und nicht 1,5 angezeigt, wie es eigentlich richtig wäre. Der Programmieraufwand zur Lösung dieser Probleme erschien zu groß im Verhältnis zum Nutzen.

Für langsame PCs nur bedingt geeignet

MS-DOS unterstützt keine Übertragungsraten über 9600 Baud. Der Grund hierfür ist sehr einfach. Der klassische IBM-PC mit 4,77 MHz Taktfrequenz konnte bei nennenswert höheren Baudraten die serielle Schnittstelle aus Geschwindigkeitsgründen

Bild 2. Das Leitungssteuerregister enthält alle wichtigen Parameter



nicht mehr schnell genug bedienen. Auch bei den heute üblichen 8-MHz-PCs bzw. bei langsamen ATs (6 MHz) treten bereits ab 19200 Baud Lesefehler auf, wenn man über die MS-DOS-Funktion (Interrupt 21H, AH=3) auf die Schnittstelle zugreift. Die BIOS-Funktion für die serielle Schnittstelle (Interrupt 14H) arbeitet erheblich schneller. Mit dem Toshiba T1200 und der BIOS-Funktion konnten unter Turbo-Pascal bei 19200 Baud längere Datenserien einwandfrei gelesen werden. Je nach Geschwindigkeit des verwendeten Rechners wird aber früher oder später auch die BIOS-Routine zu langsam. Immerhin kommt bei einer

Non-Stop-Datenübertragung mit 115 Kbaud ungefähr alle 85 µs ein Zeichen an. Hier kommt man nur noch durch direktes Abfragen per Assembler weiter.

SETCOM ist als COM-Datei entworfen worden, d. h. der Speicher für Code und Daten darf 64 KByte nicht überschreiten, alle Segmentregister werden beim Aufruf des Programms von MS-DOS auf das Codesegment eingestellt, und der Code muß im Speicher bei Offset 100H beginnen. Wenn man den in Bild 1 angegebenen Quellcode in die Datei SETCOM.ASM eingibt und mit MASM (Version 4.0) arbeitet, muß man folgende Befehlssequenz ablaufen lassen:

```
MASM SETCOM/ML/MX/Z/T,,CON;
LINK SETCOM,,CON;
EXE2BIN SETCOM.EXE SETCOM.COM
```

Denkbare Erweiterungen

Eine genauere Baudraten-Programmierung im Bereich kleiner Baudraten ließe sich durch Verwendung einer Fließkomma-Arithmetik erzielen. Vorstellbar wäre auch eine menügesteuerte Einstellung der Parameter. Die Wahl der seriellen Schnittstelle (COM1 oder COM2) könnte man durch Hinzufügen eines Parameters für die Schnittstellen-Nummer ermöglichen. Eine der wichtigsten Erweiterungen stellt die Implementierung der P-Option des Mode-Kommandos dar, die man gleich mit einem X-ON/X-OFF-Handshake verbinden könnte.

Literatur

- [1] Ray Duncan: MS-DOS für Fortgeschrittene, Vieweg Verlag, Braunschweig 1986
- [2] PC/XT/AT-Referenzliste, mc 1988, Ausgabe 2, Seite 59

Tabelle: Die höchsten Baudraten sind nur bedingt nutzbar

Baudrate	Teilerwert	
	dezimal	hexadezimal
50	2304	0900
75	1536	0600
100	1152	0480
110	1047	0417
150	768	0300
300	384	0180
600	192	00C0
1200	96	0060
1800	64	0040
2000	58	003A
2400	48	0030
3600	32	0020
4800	24	0018
7200	16	0010
9600	12	000C
10472	11	000B
11520	10	000A
12800	9	0009
14400	8	0008
16457	7	0007
19200	6	0006
23040	5	0005
28800	4	0004
38400	3	0003
57600	2	0002
115200	1	0001

Wolfgang Reese

Matrizenrechnung in C

... für Fortran-Umsteiger

Für einen Wechsel von der Programmiersprache Fortran zur Programmiersprache C müssen – vor allem für professionelle Programmierer – schon sehr gewichtige Gründe vorliegen [1]. Dennoch kommt ein solcher Wechsel in den letzten Jahren zunehmend öfter vor. Als Auslöser für den Umstieg kommt z. B. eine der folgenden Möglichkeiten in Frage:

- ☐ Wechsel des Arbeitsgebietes (Prozeßdatenverarbeitung)
- ☐ Wechsel des Rechners (für neue Rechner stehen zuerst meistens C-Compiler zur Verfügung)
- ☐ Wechsel des Betriebssystems (Unix-artige Betriebssysteme basieren auf C)
- ☐ Vorgaben durch den Auftraggeber

Nach der ersten Lektüre der „C-Bibel“ [2] stellt man überrascht fest, daß man unter C sehr komfortabel und übersichtlich programmieren kann. Wenn man will, kann man aber auch sehr umständlich und unübersichtlich programmieren. Es sieht sogar so aus, als könne man vorhandene Fortran-Programme sehr leicht in die Programmiersprache C „übersetzen“.

Man entschließt sich also zum Kauf eines C-Compilers und macht die ersten Programmierversuche. Das erste Programm (Hello, World!) funktioniert anstandslos und auf Anhieb.

Dadurch ermutigt wird das nächste Projekt etwas höher angesiedelt: Übertragung von Standard-Fortran-Unterprogrammen auf den „neuen Standard“ C.

Rein zufällig wird die Matrizen-Bearbeitung herausgegriffen und schon – Fehlermeldungen über Fehlermeldungen. Der C-Compiler will z. B. einfach nicht begreifen, daß an eine Funktion eine Matrix mit variablen Dimensionen übergeben werden soll. In der nun folgenden Programmier-Nachtschicht steigt langsam aber sicher der Frust. Nach endlosen Versuchen – test1.c, test2.c, ... – und intensivem Literaturstudium beschleicht den frisch gebackenen C-Programmierer ein Verdacht: So einfach

Matrizenrechnung in der professionellen EDV war jahrzehntelang eine Domäne der Programmiersprache Fortran. Der Beitrag schildert die Probleme, die beim Umstieg auf die Programmiersprache C auftreten, und bietet mit einem Lösungsvorschlag ein Paket von Standard-Matrizen-Funktionen in C.

wie in Fortran kann man in C mit Matrizen offenbar nicht umgehen.

Standardzugriff

Selbstverständlich kann man auch in C auf Matrizen oder mehrdimensionale Vektoren, wie sie in C genannt werden, zugreifen. Man definiert eine Matrix z. B. mit

```
float matrix[10][100];
```

Für die Reihenfolge der Indizes gilt auch in C die Eselsbrücke: Zeile zuerst, Spalte später. Zusätzlich muß der gelernte Fortran-Programmierer beachten, daß die Zeilen- und Spaltenindizes bei Null und nicht bei Eins beginnen.

Auf die so definierte Matrix kann im Verlauf des C-Programmes analog zu einem Fortran-Programm zugegriffen werden. Konstruktionen wie

```
float matrix[10][100],
      abc[100][100];
float x;
int i, j;
...
matrix[8][88] = 2.7182818;
i = 25; j = 3;
x = abc[j][i];
...
```

sind zulässig. Der Ärger fängt erst bei dem Versuch an, Matrizen an Funktionen zu übergeben, die für allgemeine Dimensionierung der Matrizen geschrieben werden sollen. So führt das Beispiel in Bild 1, mit dem in Fortran-analoger Weise eine Einheitsmatrix erzeugt werden soll, gleich zu einer ganzen Reihe von Fehlermeldungen: Die Deklaration einer Matrix als Formalparameter einer Funktion ist in C leider nur mit konstanten Dimensionen möglich. Auf

diese Weise kann man natürlich keine Funktion schreiben, die eine beliebig dimensionierte Matrix bearbeiten soll.

C ist eine Sprache, die sich zur Bearbeitung von Vektoren hervorragend eignet. Auf Vektoren kann sowohl über einen Index als auch über Zeiger sehr komfortabel zugegriffen werden.

Zudem ist in C die Definition von Vektoren zulässig, deren Komponenten Zeiger auf andere Vektoren sind. So wird denn auch folgerichtig in [2] die Verwendung von Zeigervektoren zur Lösung des Matrixproblems empfohlen.

Diese zunächst einsichtige Möglichkeit erweist sich bei näherem Hinsehen als etwas umständlich: Bevor eine Matrix nach ihrer Definition benutzt werden kann, muß zunächst ein Zeigervektor mit Zeigern auf alle Zeilenstartelemente der Matrix initialisiert werden (Bild 2).

Selbstverständlich hat dieses Vorgehen auch positive Seiten: Der Zeigervektor kann Zeiger auf unterschiedlich lange Zeilen enthalten! Leider kann man diesen Vorteil bei den meisten Problemen der Matrizenrechnung nicht ausnutzen, weil dort Matrizen mit unterschiedlich langen Zeilen höchst selten benötigt werden...

Trotz des oft erheblichen Mehraufwandes an Speicherplatz und ausführbarem Code existieren bereits Bibliotheken mit Matrix-Standardfunktionen, die auf dieser Technik basieren [3].

Fortran-ähnlicher Zugriff

Sowohl in Fortran- als auch in C-Programmen werden Matrizen intern als eindimensionale Vektoren abgespeichert. In Fortran wird dabei spaltenweise, in C zeilenweise verfahren. So wird in C die Matrix

$$a = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \\ a_{20} & a_{21} \end{pmatrix}$$

als Vektor

$$a = \{a_{00} \ a_{01} \ a_{10} \ a_{11} \ a_{20} \ a_{21}\}$$

angespeichert. Kennt man die Zeilenlänge einer Matrix, so kann man über einen Zei-

```
float abc[100][100];
einheitsmatrix(abc,100); /* Aufruf Function */

/* ----- Definition Function fehlerhaft ----- */
einheitsmatrix(matrize,laenge)
int laenge;
float matrize[laenge][laenge];
{
    int i,j;
    for(i=0;i<laenge;i++)
        for(j=0;j<laenge;j++)
            if(i != j) matrize[i][j] = 0.0;
            else matrize[i][j] = 1.0;
}
```

Bild 1. So geht es nicht: Die Definition mit float matrize bringt den Compiler zum Schimpfen

```
float abc[100][100]; /* 10000 Plaetze reserviert */
float *zabc[100]; /* 100 Zeiger reserviert */
int i;
for(i=0;i<100;i++) /* Zeiger auf Zeilenstarts */
    zabc[i] = &(abc[i][0]);

einheitsmatrix(zabc,100); /* Aufruf Function */

/* ----- Definition Function korrekt ----- */
einheitsmatrix(zmatrize,laenge)
float **zmatrize; /* oder float *(zmatrize[]) */
int laenge;
{
    int i,j;
    float *zm;
    for(i=0;i<laenge;i++)
        for(j=0,zm=zmatrize[i];j<laenge;j++,zm++)
            if(i != j) *zm = 0.0;
            else *zm = 1.0;
}
```

Bild 2. Die Lösung mit vektorisierten Zeigern: Schnell, aber speicherintensiv

```
float abc[100][100];
einheitsmatrix(abc,100); /* Aufruf Function */

/* ----- Definition Function nach FORTRAN-Art ----- */
#define A(x,y) *(a+x*n+y) /* Zugriffsdefinition a */
einheitsmatrix(a,n)
float *a;
int n;
{
    int i,j;
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            if(i != j) A(i,j) = 0.0; /* Versteckte Zugriffe */
            else A(i,j) = 1.0; /* auf Matrix a */
}
```

Bild 3. Der Algorithmus für den Fortran-ähnlichen Zugriff wurde hier in der Zeile mit #define versteckt

```
.....
* Programmname      : matrix.c
* Version           : 1.0
* Programmierer      : W.Reese
* Projekt            : Matrixbearbeitungssoftware
* Kurzbeschreibung: Matrix-Functions allgemein
.....

/*
Grundlage der Matrixbearbeitungssoftware ist die Anordnung
von Matrixelementen in C. Die Elemente werden zeilenweise
hintereinander im Speicher in einem linearen Feld angelegt.
Es ist dann moeglich, Zeiger auf Felder und deren Dimen-
sionen an Functions zu uebergeben, wenn diese nur von der
linearen Struktur gebrauch machen.
*/

#include <stdio.h>

/* ===== minv_f ===== */
/* Inversion einer Matrix mit dem Gauss-Jordan Verfahren
Das Programm wurde aus einer FORTRAN-Version auf C
umgeschrieben und nicht fuer C optimiert.
-1
AI = A

Die Matrix AI darf mit der Matrix A identisch sein.

Eingabeparameter:
a : Zeiger auf die zu invertierende Matrix A
(float *)
b : Zeiger auf Scratchfeld der Groesse n*2n
(float *)
n : Dimension der Matrix A (int)

Ausgabeparameter:
ai : Zeiger auf Platz für die invertierte Matrix AI
(float *)

Returns:
0/-1 : OK/Error

#define ABS(x) (x>0.0)?x:x*(-1.0) /* Absolutwert */
#define A(x,y) *(a+n*x+y) /* Matrixelement von a */
#define AI(x,y) *(ai+n*x+y) /* Matrixelement von ai */
#define B(x,y) *(b+n2*x+y) /* Matrixelement von b */

minv_f(a,ai,b,n)
float *a,*b,*ai;
int n;
{
    int i,j,k,l,m;
    int ni,n2;

    float h;
    n2 = n*2;

    /* ---- Kopieren a --> linke Haelfte von b
for(i=0;i<n;i++) { /* Zeilenindex 0 .. n-1
for(j=0;j<n;j++) /* Spaltenindex 0 .. n-1
    B(i,j) = A(i,j); /* b[] = a[]
}

/* ---- Einheitsmatrix --> rechte Haelfte von b
for(i=0;i<n;i++) { /* Zeilenindex 0 .. n-1
for(j=n;j<n2;j++) { /* Spaltenindex n .. 2n-1
    if((j-n) == i) /* Diagonalelem. rechte Haelfte b
        B(i,j) = 1.0; /* b[] = 1
    else /* Kein Diagonalelement rechte Haelfte b
        B(i,j) = 0.0; /* b[] = 0
}

/* ---- Loesung lin. Gleichungssystem nach Gauss-Jordan
for(i=0;i<n;i++) { /* Zeilen-Index = 0..n-1
/* ---- Fuer alle Zeilen
j = i; /* Zeilenindex=Spaltenindex=Index

/* ----- Zeilentauch damit das absolut groesste Element
der Spalte j in der Zeile i zu liegen kommt
h = ABS(B(i,j)); m = j;
for(k=i+1;k<n;k++) { /* Naechste Zeile..Endzeile
    if(ABS(B(k,j)) > h) {
        h = ABS(B(k,j));
        m = k;
    }
}
if(h < 1.0E-8) /* Alle Elemente = 0
return(-1); /* Matrix singular
if(m != i) { /* Zeilentauch noetig
for(k=j;k<n2;k++) {
    h = B(i,k);
    B(i,k) = B(m,k);
}
```

Bild 4. Eine Sammlung von Matrix-Standard-Functionen

```

    B(m,k) = h;
}

/* ----- Jordan-Schritt durchfuehren
for(k=0;k<n;k++) { /* Alle Zeilen
    if(k != i) {
        h = B(k,j)/B(i,j);
        for(m=j;m<n2;m++)
            B(k,m) = B(k,m)-h*B(i,m);
    }
}

/* ----- Normalisieren der Jordan-Form
for(i=0;i<n;i++) { /* Alle Zeilen

    h = B(i,i);
    for(j=n;j<n2;j++) /* Spalten n..n-1
        B(i,j) = B(i,j)/h;
}

/* ----- Kopieren rechte Haelfte b --> ai
for(i=0;i<n;i++) { /* Zeilenindex 0 .. n-1
    for(j=0;j<n;j++) { /* Spaltenindex 0 .. n-1
        ni = n+j; /* Spaltenindex n .. 2n-1
        AI(i,j) = B(i,ni); /* a[] = b[]
    }
}

return(0); /* Matrix invertiert

/* ===== minv_c =====
/* Inversion einer Matrix mit dem Gauss-Jordan Verfahren.
Entwickelt aus einem FORTRAN-Programm und optimiert
fuer C

    -1
    AI = A

Die Matrix AI darf mit der Matrix A identisch sein.

Eingabeparameter:
    a : Zeiger auf die zu invertierende Matrix A
        (float *)
    n : Dimension der Matrix A (int)

Ausgabeparameter:
    ai : Zeiger auf Platz fuer die invertierte Matrix AI
        (float *)

Returns:
    0/-1 : OK/Matrix singular
    -2 : Nicht genug Speicher fuer Hilfsfeld frei

#define ABS(x) ((x)>0.0)?x:(-1.0) /* Absolutwert */

minv_c(a,ai,n)
float *a,*ai;
int n;
{
    register int i,k,m;
    int n2;
    float h,hh;
    float *b,*zeile,*spalte,*diag;
    float *sa,*zb;
    char *malloc();

    /* ----- Speicher fuer Hilfsfeld b anfordern
    n2 = n*2;

    i = n2*n*sizeof(h); /* Speichergroesse Hilfsfeld n2*n
    if((b=(float *)malloc(i)) == (float *)0)
        return(-2); /* Error zu wenig Speicher

    /* ----- Kopieren a --> linke Haelfte von b,
    b --> rechte Haelfte von b
    for(i=0;zeile=b;i<n;i++) { /* Alle Zeilen
        for(m=0;m<n;m++) /* Alle Spalten von a
            *zeile++ = *a++; /* Zeile von a kopieren
        for(k=0;k<n;k++) /* Alle Spalten Einheitsmatrix
            if(k != i) *zeile++ = 0.0;
            else *zeile++ = 1.0;
    }

    /* ----- Loesung lin. Gleichungssystem nach Gauss-Jordan
    for(i=0;zeile=b,spalte=b;i<n;
        i++,zeile+=n2,spalte++) { /* Zeilen-Index = 0..n-1
        /* ----- Fuer alle Zeilen
        diag = zeile+i; /* Zeiger auf Diagonalelement

        /* ----- Zeilentausch damit das absolut groesste Element
        der Spalte i in der Zeile i zu liegen kommt
        za = diag; /* Zeiger auf Startelement der Suche

```

```

        h = ABS(*za); m = i;
        za += n2;
        for(k=i+1;k<n;k++,za+=n2) { /* Naechste Zeile..Endzeile */
            if((hh=ABS(*za)) > h) {
                h = hh;
                m = k;
            }
            if(h < 1.0E-8) { /* Alle Elemente = 0
                free((char *)b); /* Speicher zurueckgeben
                return(-1); /* Matrix singular

            if(m != i) { /* Zeilentausch noetig
                za = zeile+i; zb = spalte+m*n2;
                for(k=i;k<n2;k++) {
                    h = *za;
                    *za++ = *zb;
                    *zb++ = h;
                }
            }

            /* ----- Jordan-Schritt durchfuehren
            for(k=0;k<n;k++) { /* Alle Zeilen
                if(k != i) {
                    za = spalte+k*n2;
                    zb = diag;
                    h = (*za)/(*zb);
                    for(m=i;m<n2;m++)
                        (*za++) -= h*(*zb++);
                }
            }

            /* ----- Normalisieren der Jordan-Form und --> ai
            for(i=0;diag=b,za=b+n2;i<n;
                i++,diag+=n2+1,za+=n) { /* Alle Zeilen
                for(k=0;k<n;k++)
                    *ai++ = (*za++)/(*diag);
            }

            free((char *)b); /* Speicher zurueckgeben
            return(0); /* Matrix invertiert

            /* ===== pivot =====
            /* Loesung eines linearen, inhomogenen Gleichungssystems
            mit dem Pivotisierungsverfahren.

            A*X = b, A[n,n], X[n,1], B[n,1]

            Leicht modifiziert aus:
            Vogeler, J.-H., Approximationspolynome in C,
            Zeitschrift NC 12/87, Dezember 1987.

            Eingangsdaten:
                a : Zeiger auf die Koeffizientenmatrix A
                    (float *)
                b : Zeiger auf den Vektor der rechten Seite B
                    (float *)
                n : Ordnung des Systems (int)

            Ausgangsdaten:
                a : Zeiger auf invertierte Koeffizientenmatrix
                    (float *)
                b : Zeiger auf Loesungsvektor (float *)
                determ : Zeiger auf Determinante der Koeffizienten-
                    matrix (float *)

            Returns 0/(-1): OK/Error

            #define swap(x,y) {temp=x;x=y;y=temp;}
            #define fabs(x) ((x)>0.0)?x:(-1.0):x
            #define MAXORD 20

            pivot(a,b,n,determ)
            float *a,*b,*determ;
            int n;
            {
                register int i,j,k,l,m;
                int indrow[MAXORD],irow, /* Speicher fuer Zeilenindex
                indcol[MAXORD],icol, /* Speicher fuer Zeilenindex
                ipivot[MAXORD]; /* Speicher fuer Pivotflag
                float pivot,temp;

                if((n<1) || (n)>MAXORD) /* Fehler Systemordnung
                    return(-1); /* Gleichungssystem nicht loesbar

                for(i=0;i<MAXORD;i++) {
                    indrow[i] = 0;
                    indcol[i] = 0;
                    ipivot[i] = 0;
                }
                *determ = 1.0;

```

```

for(i=0;i<n;i++) {
    temp = 0.0;
    for(j=0;j<n;j++) {
        for(k=0;k<n;k++) {
            /* ----- Betragsgrösstes Pivotelement suchen */
            if((!(ipivot+j)!=1)&&(k<n):k++) {
                /* ----- */
                if((!(ipivot+k)<1) && (fabs(temp)<fabs(a[n+j+k]))) {
                    irow = j;
                    icol = k;
                    temp = a[n+j+k];
                }
            }
        }
    }

    if(temp == 0.0) /* Koeffizientenmatrix singulaer */
        return(-1); /* Gleichungssystem nicht loesbar */

    if(irow != icol) {
        /*determ *= (-1.0);
        for(l=0;l<n;l++)
            swap(b[irow],b[icol]);*/

        /*(irow+1) = irow;
        /*(icol+1) = icol;
        /*(ipivot+icol) = 1;
        /*(ipivot+icol) = 1;
        pivot = a[icol*(n+1)];
        /*determ *= pivot;
        /*determ *= pivot;
        a[icol*(n+1)] = 1.0;
        for(l=0;l<n;l++)
            a[icol*(n+1)] /= pivot;
        b[icol] /= pivot;
        for(m=0;m<n;m++) {
            if(m == icol)
                continue;
            temp = a[n*m+icol];
            a[n*m+icol] = 0.0;
            for(l=0;l<n;l++)
                a[n*m+1] -= a[n*icol+1]*temp;
            b[m] -= b[icol]*temp;
        }

        for(i=n-1;i>=0;i--) {
            if((irow==(indrow+1)) == (icol==(indcol+1)))
                continue;
            for(k=0;k<n;k++)
                swap(a[n*k+irow],a[n*k+icol]);
        }

        return(0); /* Gleichungssystem geloest */

/* ----- matprint ----- */
/* Ausgabe einer Matrix auf STDOUT zu Testzwecken

    Eingabeparameter:
        a : Zeiger auf die Matrix (float *)
        n : Anzahl der Zeilen (int)
        m : Anzahl der Spalten (int)

matprint(a,n,m)
float *a;
int n,m;
{
    int i,j;
    for(i=0;i<n;i++) { /* Zeilen der Matrix
        printf("\n");
        for(j=0;j<m;j++) /* Spalten der Matrix
            printf("%5.2g ",a[i+j]); /* Ausgabe Matrixelement
        printf("\n");
    }
    fflush(stdout);
}

/* ----- matmul ----- */
/* Multiplikation von zwei Matrizen C = A*B
        A[l,m]
        B[m,n]
        C[l,n]

    Eingabedaten:
        a : Zeiger auf Matrix A (float *)
        b : Zeiger auf Matrix B (float *)
        l : Anzahl der Zeilen Matrix A und Anzahl der
            Zeilen B (int)
        m : Anzahl der Spalten A und Anzahl der
            Zeilen B (int)
        n : Anzahl der Spalten B und Anzahl der
            Spalten C (int)

    Ausgabedaten:
        c : Zeiger auf Matrix C (float *)

matmul(a,b,c,l,m,n)
float *a,*b,*c;
int l,m,n;
{
    float *pa,*pb,x;
    register int i,j,k;

    for(i=0;i<l;i++) { /* Alle Zeilen von A
        for(j=0;j<n;j++) { /* Alle Spalten von B
            pa = a+i*m; /* Zeiger Start Zeile A
            pb = b+j; /* Zeiger Start Spalte B
            x = 0.0; /* Kummulierungsvariable = 0
            for(k=0;k<m;k++) { /* Alle Spalten von A bzw. alle
                /* Zeilen v. B
                x += (*pa)*(*pb);
                pa++;
                pb += n;
            }
            *c++ = x; /* Speichern in Matrix C
        }
    }
    return(0); /* Exit Status immer OK
}

/* ----- transpon ----- */
/* Transponieren einer Matrix
        B = A
        A[l,m] B[m,l]

    Eingabedaten:
        a = Zeiger auf Matrix A (float *)
        l = Zeilenzahl von A (int) = Spaltenzahl von B
        m = Spaltenzahl von A (int) = Zeilenzahl von B

    Ausgabedaten:
        b = Zeiger auf Matrix B (float *)

transpon(a,b,l,m)
float *a,*b;
int l,m;
{
    register int i,j;
    float *pb;
    for(i=0;i<l;i++) { /* Alle Zeilen von A
        /* ----- Zeile von A auf Spalte von B unspeichern
        pb = b+i; /* Zeiger auf Start der Spalte von B
        for(j=0;j<m;j++) { /* Ganze Zeile von A
            /* Element unspeichern
            *pb = *a++;
            pb += l;
        }
    }
    return(0);
}

/* ----- matadd ----- */
/* Addition von zwei Matrizen
        C = A+B
    Matrix C darf mit Matrix A oder B identisch sein

    Eingabedaten:
        a = Zeiger auf Matrix A (float *)
        b = Zeiger auf Matrix B (float *)
        l = Zeilenzahl von A,B,C (int)
        m = Spaltenzahl von A,B,C (int)

    Ausgabedaten:
        c = Zeiger auf Matrix C (float *)

matadd(a,b,c,l,m)
float *a,*b,*c;
int l,m;
{
    register int i,j;
    j = l*m; /* Anzahl der Matrixelemente
    for(i=0;i<j;i++) /* Alle Matrixelemente
        *c++ = (*a++)+(*b++); /* Addition zweier gleichstehender
        /* Elemente
    }
    return(0);
}

/* ----- matsub ----- */
/* Subtraktion von zwei Matrizen
        C = A-B
    Matrix C darf mit Matrix A oder B identisch sein

    Eingabedaten:
        a = Zeiger auf Matrix A (float *)
        b = Zeiger auf Matrix B (float *)
        l = Zeilenzahl von A,B,C (int)
        m = Spaltenzahl von A,B,C (int)

    Ausgabedaten:
        c = Zeiger auf Matrix C (float *)

matsub(a,b,c,l,m)
float *a,*b,*c;
int l,m;
{

```



```

register int i,j;
j = 1*m; /* Anzahl der Matrixelemente */
for(i=0;i<j;i++) /* Alle Matrixelemente */
    *a++ = (*a++)-(b++); /* Subtraktion zweier gleich-
                           stehender Elemente */
return(0);

/* ===== matfakt ===== */
/* Multiplikation einer Matrix mit einem Faktor
   B = A*f
   Die Matrix A darf mit der Matrix B identisch sein

Eingabedaten:
a = Zeiger auf die Matrix A
l = Zeilenzahl der Matrix A und B (int)
m = Spaltenzahl der Matrix A und B (int)
f = Faktor (float)

Ausgabedaten:
b = Zeiger auf Matrix B (float *)

matfakt(a,b,l,m,f)
float *a,*b,f;
int l,m;
{
register int i,j;
j = 1*m; /* Anzahl der Matrixelemente */
for(i=0;i<j;i++) /* Alle Matrixelemente */
    *b++ = f*(a++); /* Multiplikation eines Elementes */
return(0);

/* ===== mateinh ===== */
/* Laden einer quadratischen Einheitsmatrix
   A = E

```

```

Eingabedaten:
l = Zeilen/Spaltenzahl von A (int)

Ausgabedaten:
a = Zeiger auf die Matrix A (float *)

mateinh(a,l)
float *a;
int l;
{
register int i,j;
*a++ = 1.0; /* A[0][0] = 1 */
for(i=1;i<l;i++) {
    for(j=0;j<l;j++) *a++ = 0.0;
    *a++ = 1.0;
}
return(0);

/* ===== matcopy ===== */
/* Kopieren einer Matrix
   B = A

Eingabeparameter:
a : Zeiger auf die Matrix A (float *)
l : Zeilenzahl von A und B (int)
m : Spaltenzahl von A und B (int)

Ausgabedaten:
b : Zeiger auf die Matrix B (float *)

matcopy(a,b,l,m)
float *a,*b;
int l,m;
{
register int i,j;
j = 1*m; /* Anzahl der Matrixelemente */
for(i=0;i<j;i++) /* Alle Matrixelemente */
    *b++ = *a++; /* Matrixelement kopieren */
return(0);
}

```

ger auf den Start der Matrix und einen berechneten Offset auf jedes Element der Matrix zugreifen:

Zeiger(a/x|y|)
 = Zeiger(a) + x * Zeilenlänge + y

Sowohl ein Zeiger auf den Start einer Matrix als auch die Zeilenlänge können problemlos an eine Funktion übergeben werden. Innerhalb der Funktion werden dann alle Zugriffe auf die Matrix nach der eben beschriebenen Methode über Zeiger und berechneten Offset durchgeführt. Ein solcher Zugriff auf ein Matrixelement ist natürlich nicht sehr elegant. Im Listing eines Programmes würde diese etwas unübersichtliche Zugriffsart sogar sicherlich Verwirrung stiften. Die Lesbarkeit wird wesentlich verbessert, wenn man den eigentlichen Zugriffsalgorithmus versteckt. Dazu bietet sich der Präprozessor an (Bild 3).

Performance

Geschwindigkeit und Speicherverbrauch (performance) hängen nicht unwesentlich von der Art des Zugriffs auf die Matrizen ab. Manchmal ist aber auch die Programm-Entwicklungszeit der entscheidende Faktor – dies ist vor allem im Hochschulbereich häufiger anzutreffen: Hier werden mit hohem Zeitaufwand Programmentwicklungen betrieben; die fertigen Programme laufen dann nur einige Male, bis der Programm-

entwickler sein kurzfristiges Ziel (Diplom, Veröffentlichung usw.) erreicht hat. Der Fortran-ähnliche Zugriff kann die Geschwindigkeit eines C-Programmes im Vergleich zu einem normalen Fortran-Programm kaum steigern, da die Matrix-Zugriffe in Fortran ähnlich berechnet werden. Die Programme benötigen aber im allgemeinen weniger Speicherplatz als entsprechende Programme mit Standard-C-Zugriffen. Der Standard-C-Zugriff auf Matrixelemente führt normalerweise immer zu schnelleren Programmen, da das Laden der Vektoren mit den Zeigern auf die Zeilenstartelemente normalerweise nur einmal durchgeführt wird, beim Zugriff auf einzelne Matrixelemente aber damit die Multiplikation mit der Zeilenlänge wegfällt. Dies wirkt sich vor allem dann aus, wenn in Programmschleifen sehr häufig auf einzelne Matrixelemente zugegriffen wird.

Die Umsetzung eines Fortran-Programmes in ein C-Programm ist mit der Fortran-ähnlichen Zugriffsmethode wesentlich schneller durchzuführen. Nachteilig ist dabei, daß der Geschwindigkeitsvorteil eines guten C-Programms im Vergleich zu einem guten Fortran-Programm nicht genutzt wird. Vor allem beim Programmieren komplexerer Matrix-Standard-Funktionen (etwa Inversion) sollte man von den schnellen C-Operationen wie In- und Dekrementierung in Programmschleifen Gebrauch machen.

In Bild 4 sind Matrix-Standard-Funktionen für häufig gebrauchte Matrix-Operationen. Für die Inversion wurde dabei die Funktion `minv_f` mit Fortran-ähnlichem Zugriff und die auf dem selben Algorithmus [4, Seite 93...95] basierende Funktion `minv_c` mit Standard-C-Zugriff realisiert.

Ein Geschwindigkeitsvergleich auf einem Rechner mit der CPU 68000 ohne Arithmetik-Prozessor ergibt einen Geschwindigkeitsvorteil von rund 10 % für `minv_c`. Die Verbesserung fällt relativ gering aus, weil die Fließkomma-Operationen hierbei den größten Teil der Rechenzeit benötigen. Mit Arithmetik-Prozessor sinken die Ausführungszeiten für Fließkomma-Operationen auf die Größenordnung derjenigen für Festkomma-Operationen, der relative Geschwindigkeitsvorteil zugriffsoptimierter C-Programme wird dann wesentlich stärker.

Literatur:

- [1] F. Lüther, Th. v. Wirth: Der Stoff aus dem die Software ist, Computer Persönlich, 1988, Heft 8, Seite 54
- [2] Brian W. Kernighan, Dennis M. Ritchie: Programmieren in C, Hanser Verlag, München/Wien, 1983
- [3] G. Engeln-Müllges, F. Reutter: Formelsammlung zur numerischen Mathematik mit C-Programmen, BI-Verlag, Mannheim/Wien/Zürich, 1987
- [4] G. Jordan-Engeln, F. Reutter: Numerische Mathematik für Ingenieure, BI-Verlag, Mannheim/Wien/Zürich, 1978

Thomas Adler, Ingrid Trommer

LAN-Technik und uucp

Teil 2: Kommunikation unter Unix

An der Tatsache, daß das Programmpaket uucp standardmäßig auf jedem Unix-Rechner vorhanden ist, kann man sofort sehen, daß das Betriebssystem Unix ursprünglich von einer Telefon-Firma (AT&T) entwickelt wurde: Mit dieser Software nämlich kann man mit einer einfachen seriellen Schnittstelle (V.24) oder einem Telefonmodem mit einem anderen Computersystem kommunizieren.

Kommunikation innerhalb eines Systems

Hier gibt es zwei Möglichkeiten, nämlich „mail“ und „write“. Man kann jedem Anwender, der einen Eintrag in der Datei „/etc/passwd“ hat, eine Nachricht über „mail“ zukommen lassen. Wenn sich der „Empfänger“ das nächste Mal einloggt, erhält er die Meldung „You have mail“. Bild 8 zeigt nacheinander das Senden und Empfangen von „mail“.

Mit „write“ kann man mit jedem Anwender kommunizieren, der momentan eingeloggt ist. Man kann ihm z. B. mitten in eine Editorsitzung hineinspucken. Das ist bei Unix-Kursen ein beliebter Sport. Allerdings ist es auch möglich, sich gegen ungewollte Nachrichten zu schützen. Hierfür findet der Befehl „mesg n“ Verwendung. Er nimmt die Schreibberechtigung für alle anderen von der eigenen Terminalleitung weg, und man hat Ruhe. Die Syntax von „write“ ist in Bild 9 erklärt.

uucp entstand zu einem Zeitpunkt, als über Datenkommunikation nur an wenigen Stellen nachgedacht wurde. Das Paket ist aus den Erfordernissen eines großen Instituts, den Bell-Laboratorien, entstanden. Standards gab es damals noch nicht, so daß es nicht möglich ist, eine Verbindung zu dem OSI-7-Schichten-Modell herzustellen. Da aber die Entwickler von uucp sich an den wirklich vorhandenen Bedürfnissen orientierten, ist das Paket von der Funktionalität her durchaus mit den inzwischen entstandenen Standards vergleichbar.

Es besteht aus zwei logisch voneinander getrennten Teilen:

Nachdem im ersten Teil einige Grundlagen über Netze und, darauf aufbauend, mehrere Protokolle vorgestellt wurden, geht es diesmal um eine standardmäßig auf allen Unix-Rechnern vorhandene, also weitverbreitete Kommunikationssoftware: Das Programm „Unix to Unix Copy“ – kurz „uucp“.

– Der Teil zur Kommunikation mit Rechnern außerhalb der Unix-Welt wird cu genannt

– uucp heißt der Teil zur Kommunikation mit Unix-Rechnern

Kommunikation von Unix zu anderen Betriebssystemen

cu ist ein asynchroner Terminalemulator. Das Unix-System muß mit der Fremdmaschine über eine serielle, asynchrone Schnittstelle verbunden sein. Die Fremdmaschine muß in der Lage sein, ein asynchrones Terminal zu unterstützen. Außerdem ist ein gemeinsames Handshake-Protokoll (xon/xoff, enq/ack) und die gleiche Baudrate auf beiden Seiten notwendig. Mit dem Aufruf von cu lassen sich diese Optionen spezifizieren.

Mit cu kann man sich auf der Fremdmaschine einloggen. Dazu muß auf der anderen Maschine ein Prozeß auf der entsprechenden Leitung aufgesetzt sein, der dem „getty“-Prozeß (der getty-Prozeß setzt den Terminal-Port entsprechend den Terminal-Charakteristika und stößt die Login-Prozedur an) auf dem Unix-System entspricht. Das Unix-System verhält sich dann so, als wäre es ein Terminal an der anderen Maschine. Kennt man die Syntax des anderen Betriebssystems, kann man dort Prozesse starten und so arbeiten, als wäre das Terminal direkt am Fremdrechner angeschlossen. Natürlich muß man an dem Fremdrechner auch eine gültige Benutzerkennung besitzen.

Selbstverständlich kann der Anwender auch Text- oder Quelldateien zu und von der Fremdmaschine übertragen. Will man eine Datei von der Fremdmaschine erhalten, muß dort ein Sendeprozess gestartet werden. Will man der Fremdmaschine eine Datei schicken, so muß dort ein Empfangsprozess laufen. Obwohl die geschilderte

Prozedur etwas umständlich ist, arbeitet man mit einem gesicherten Protokoll, und der Übertragungsweg ist kostengünstig.

Unix-zu-Unix-Kommunikation

Will man mit einem anderen Unix-System kommunizieren,

geht das bei weitem komfortabler.

An die Anwender des anderen Unix-Systems kann Post geschickt werden. Die Syntax ist dieselbe, wie innerhalb des eigenen Systems, man muß nur den Namen des Fremdsystems hinzufügen. Man kann sowohl Text- oder Quelldateien als auch Binärdateien übertragen. Auf dem Fremdsystem kann man Prozesse und Batch-Prozesse starten. Es ist sogar möglich, durch eine ganze Reihe von Fremdsystemen mit einem endlichen Zielsystem zu kommunizieren. Dabei „weiß“ jeder der beteiligten Rechner, an welches System er die entsprechenden Daten weiterzureichen hat. Den Weg vom Sendesystem zum Empfangssystem muß dabei der Absender vorschreiben. Ein Rechner innerhalb der Kette reicht die Daten dann an den jeweils nächsten Rechner weiter.

Für alle diese Funktionen braucht der Benutzer nur den oder die Namen der beteiligten Systeme zu kennen. Er muß die Berechtigung haben, auf die entsprechenden Daten zuzugreifen. Das eventuell notwendige Anwählen der Maschine, wenn es sich nicht um eine Direktverbindung handelt, das Starten von Prozessen auf dieser Maschine oder die Zwischenlagerung von Daten in Spool-Dateien erledigt das Paket „uucp“. Um diese Funktionen zu ermöglichen, muß auf der eigenen und auf der Fremdmaschine zunächst einmal eine ganze Reihe von Dateien durch den Systemadministrator richtig aufgesetzt sein.

Bei uucp erfolgt die Kommunikation grundsätzlich über eine V.24-Schnittstelle. Die Verbindung kann eine direkte Leitung sein. Die Übertragungsgeschwindigkeit liegt dann typischerweise bei 9600 oder 19200 Baud. Es kann auch eine Modem-Verbindung geschaltet werden, deren Geschwindigkeit zwischen 300 und 4800 Baud liegt. Das Modem wandelt digitale Signale in ana-

GRUNDLAGEN

```
$
$
$ mail hugo
Achtung, Meeting am 12.1.88 muss um eine Woche verschoben werden.
Neuer Termin Dienstag den 19.1.88 um 9.00 Uhr.
Gruss Ingrid.
$
$
$ logout

login: hugo
Erase set to Backspace
(c) Copyright 1983, 1984, 1985 Hewlett-Packard Co.
(c) Copyright 1979 The Regents of the University of Colorado, a body corporate
(c) Copyright 1979, 1980, 1983 The Regents of the University of California
(c) Copyright 1980, 1984 AT&T Technologies. All Rights Reserved.

Welcome to Hewlett-Packard System 9000 HP-UX
You have mail.
$ mail
From ingrid Sun Jan 10 13:16:12 1988
Achtung, Meeting am 12.1.88 muss um eine Woche verschoben werden.
Neuer Termin Dienstag den 19.1.88 um 9.00 Uhr.
Gruss Ingrid.

? n
$
```

Bild 8. So schickt man unter Unix eine Nachricht über ein LAN

```
? n
$
$
$
$
$ write ingrid
ingrid is not logged on.
$ who
hugo      console      Jan 10 13:18
thomas    tty01        Jan 10 13:24
$ write thomas
Heute 17.00 Uhr Besprechung.
$
    Message from thomas (tty01) [ Sun Jan 10 13:28:02 ] ...
ok. thomas
<EOT>
```

Bild 9. Ein Beispiel für den Befehl write

loge Signale um, die über das Fernsprechnetz übertragbar sind. Mit einem Modem kann man auch entfernte Rechner über X.25, also Datex-P bzw. Datex-L erreichen. Der Verbindungsaufbau wird hier in drei Abschnitte unterteilt. Der erste Abschnitt ist der Weg vom eigenen Modem zum örtlichen PAD (packet assembly disassembly), dann weiter zum entfernten PAD, von da zum Modem des Empfängers. Die PADs setzen die vom Modem gesendeten analogen Signale in die Datenpakete des X.25-Netzes um.

Übertragungsprotokoll

Gibt ein Benutzer an einem Unix-System den Befehl, daß eine Datei zu einem anderen Unix-System übertragen werden soll, initiiert er damit, von ihm selbst unbemerkt, die folgenden Schritte: Zunächst einmal wird das Kommando in ein Spool-Directory geschrieben. Dann wird ein weiterer Prozeß aufgerufen, der

sich aus einer Systemdatei die Information holt, wie er das Fremdsystem erreichen kann. Danach wird überprüft, ob auf der gewünschten Leitung gerade ein Transfer stattfindet. Wenn nein, wird eine sogenannte Lockfile, eine Sperrdatei, auf beiden Systemen angelegt, um zu verhindern, daß weitere Anwender auf dieser Leitung einen Transfer starten. Der Prozeß wählt sich auf dem Fremdsystem mit seiner Benutzerkennung ein. Damit startet er denselben Prozeß auf dem Fremdsystem. Die beiden Prozesse tauschen eine Reihe von Nachrichten aus, um den richtigen Handshake und die Baudrate festzulegen. Nun können die Daten übertragen werden. Sie werden in Pakete zerlegt, die eine Länge von 64 Bit haben. Jedes der Pakete enthält eine Prüfsumme, um sicher sein zu können, daß eine korrekte Übertragung stattgefunden hat. Wird anhand der Prüfsumme festgestellt, daß die Übertragung nicht korrekt war, wird sie für das fehlerhafte Paket bis zu fünfmal wiederholt. Wird dann festge-

stellt, daß die Übertragung immer noch nicht erfolgreich war, wird die Verbindung abgebrochen. Die Datei muß dann noch einmal übertragen werden. Nach einer erfolgreichen Übertragung versucht der initiiierende Prozeß in seinem Spool-Directory weitere Aufträge zu finden, die für das System gelten, mit dem die Verbindung aufgebaut ist. Sind keine weiteren vorhanden, geht eine Anfrage an das andere System, ob es in Ordnung ist, die Verbindung abzubrechen. Das Fremdsystem versucht dann in seinem Spool-Directory Aufträge für das initiiierende System zu finden. Wenn solche vorhanden sind, werden sie ausgeführt. Wenn nicht, werden die Sperrdateien entfernt, und die Verbindung wird abgebrochen.

Es wurde weiter oben schon gesagt, daß der System-Administrator bestimmte Dateien einrichten muß, damit uucp fähig ist, seine Aufgaben durchzuführen. Der Inhalt dieser Dateien soll im folgenden kurz besprochen werden.

Es gibt eine Datei, in der die Information darüber gespeichert ist, zu welchen Systemen eine Verbindung besteht, zu welchen Zeiten eine Verbindung mit diesen Systemen aufgebaut werden kann, ob eine direkte oder eine Modem-Verbindung besteht, die Baudrate der Verbindung und wie man sich in das Fremdsystem einwählt. In einer weiteren Datei wird der Name der Leitung spezifiziert, wie er in dem Directory /dev zu finden ist, des weiteren der Typ der Verbindung, entweder direkt oder der Name des verwendeten Modems und die Baudrate. In einer Datei kann der System-Administrator spezifizieren, welche Befehle ein Fremdsystem im eigenen System ausführen berechtigt ist. □

Harald Gosebruch

Der Signal-EMUF

Teil 3: Programmierung

Im Rahmen dieses Beitrages kann die Befehls-Struktur des TMS 32010 und die Funktionsweise der einzelnen Register dieses Signalprozessors natürlich nur angerissen werden. Der Programmierer benötigt auf jeden Fall den von Texas Instruments angebotenen „User's Guide“ [2], in dem sämtliche Details recht gut erklärt sind. Für dieses Handbuch muß man rund 30 DM veranschlagen.

Die Register

Wie in jedem Mikroprozessor stehen auch hier unterschiedliche Register und Datenspeicher zur Verfügung:

ACC	Akkumulator (32 Bit)
AR0/1	16-Bit-Register zur indirekten Adressierung und für Schleifenzähler
ARP	Dieses Flag-Register (1 Bit) dient der Auswahl des aktuellen ARx-Registers
DP	Das Ein-Bit-Register selektiert die aktuelle Seite des Datenspeichers (1 Seite = 128 Worte)
INTF	Ein Flag-Register, das anzeigt, ob ein Interrupt aufgetreten ist.
INTM	Mit diesem Bit läßt sich der Interrupt maskieren.
OV	Dieses Bit zeigt einen arithmetischen Überlauf an.
OVN	Ein Bit zeigt den Modus der arithmetischen Einheit an.
P	Das Ergebnis einer Multiplikation wird in diesem 32-Bit-Register abgelegt.
T	Ein 16 Bit breites Register, daß den Multiplikant bei einer Multiplikation enthält.

Über diese Arbeitsregister hinaus verfügt der TMS 32010 über 144 Worte (16 Bit) Datenspeicher (D), die, wie aus dem Befehlssatz ersichtlich, von den meisten Befehlen fast schon wie der Akkumulator an-

Nachdem im zweiten Teil ein Einplatinencomputer mit einer Leistung von bis zu 5 Millionen Befehlen pro Sekunde (auch Multiplikation) aufgebaut wurde, wird in diesem vorläufig letzten Teil die Software-Erstellung beleuchtet: Der Befehlssatz des TMS 32010 und die Programmierung.

Adresse	Befehl	HEX-Code	Kommentar
000	B 010	F9 00	Springt bei RESET zur
001		00 10	Initialisierung in Adresse 010
Interrupt-Routine			
002	IN 0,3	43 00	Lade Wert vom A/D Wandler in
003	NOP	7F 80	Datenadresse 0.
004	OUT 0,3	4B 00	Sicherheit für I/O Bedienung
005	NOP	7F 80	Ausgabe des Wertes in Daten-
006	NOP	7F 80	adresse 0 an den D/A Wandler
007	NOP	7F 80	Sicherheit für I/O Bedienung
008	IN 0,6	46 00	Einlesen des binären Einganges
009	OUT 0,6	4E 00	in Datenadresse 0
00A	NOP	7F 80	Ausgabe von Datenadresse 0 an
00B	NOP	7F 80	binären Ausgang
00C	OUT 0,7	4F 00	Sicherheit für I/O Bedienung
00D	EINT	7F 82	Ausgabe von Datenadresse 0 an
00E	B 00D	F9 00	das Eingangswahl-Flip-Flop
00F		00 0D	(nur Bit 0 ist entscheidend)
			Ermögliche Interrupt
			Warteschleife für Interrupt
			springt zu Adresse 00D
010	IN 0,6	46 00	Einlesen des binären Einganges
011	OUT 0,6	4E 00	in Datenadresse 0, Beginn der
012	OUT 0,5	4D 00	Initialisierungsroutine nach
013	B 002	F9 00	RESET
014		00 02	Ausgabe von Datenadresse 0 an
			binären Ausgang
			Ausgabe von Datenadresse 0 an
			den programmierbaren Zähler
			Ende der Initialisierung,
			Springung zu Adresse 002

Bild 1. Mit 21 Programmzeilen werden hier alle Schnittstellen und der Interrupt getestet

gesprochen werden können. Zwei Links-Schieberegister (S) ermöglichen Schiebeoperationen von Datenworten eingebettet in andere Befehle und ein 4 X 12-Bit-Stack steht für Unterprogrammaufrufe zur Verfügung, es können also Unterprogramme in maximal vier Ebenen geschachtelt werden. In Assemblerprogrammen finden noch weitere Zeiger Verwendung:

K, I	definieren den Adressmodus, in dem der Befehl ausgeführt wird.
PA	PA0...PA7 entsprechen den Ein-/Ausgabe-Schnittstellen 0...7
D	Eines der RAM-Register (Im Signal-EMUF 144 Worte).

Programme und Daten liegen grundsätzlich im 16-Bit-Format vor. Auf dem Signal-EMUF sind daher zwei EPROMs vorhanden: EPROM IC03 enthält die unteren (D0...D7) und EPROM IC04 die oberen 8 Bit (D8...D15) des Programmcodes. Die Tabelle zeigt den Befehlssatz des TMS 32010.

Am Beispiel eines Testprogramms (Bild 1) soll demonstriert werden, wie die Schnittstellen und der Interrupt des Signal-EMUFs genutzt werden können. Nach einem System-Reset wird eine Konstante (8 Bit) von der parallelen TTL-Schnittstelle eingelesen, zur Kontrolle an die entsprechenden TTL-Ausgänge geschrieben und zur Programmierung des Zählers verwendet. Der Zähler bestimmt die Frequenz zum Einlesen der analogen Werte mit dem A/D-Wandler. Nach der Initialisierung erfolgt ein Sprung in die Interrupt-Routine, die den vom Auffang-Register des A/D-Wandlers eingelesenen Wert unverändert an den D/A-Wandler ausgibt. Anschließend wird erneut der 8-Bit-TTL-Eingang gelesen, zur Kontrolle ausgegeben und mit Bit 0 dieses Wertes das Eingang-Flip-Flop gesetzt. So lassen sich über die TTL-Schnitt-

stelle Wandelfrequenz und einer der beiden Analog-Eingänge auswählen. In einer Programmschleife wartet der Prozessor anschließend auf den nächsten Interrupt, der vom A/D-Wandler nach erfolgter Analogwertumsetzung ausgelöst wird. Bei einem Interrupt springt der Prozessor grundsätzlich zur Programmadresse 002.

Interrupts

Ein Interrupt wird ausgelöst, wenn der INT-Eingang des TMS 32010 auf den 'low'-Zustand wechselt. Sobald ein Interrupt erkannt wird, werden weitere Interruptanforderungen abgeblockt (Interrupt disable).

GRUNDLAGEN

Nach Abarbeitung eines Interrupts muß daher durch die EINT-Anweisung der Interrupt wieder freigegeben werden. Der Prozessor reagiert nicht auf die fallende Flanke des IRQ-Signals, sondern auf dessen Zustand: Ist das Signal, daß den Interrupt erzeugt hat, bei Ausführung der EINT-Anweisung noch nicht wieder in den 'high'-Zustand zurückgekehrt, so wird ein zweiter Interrupt erzeugt. Auf dem Signal-EMUF ist eine Mindestzeit von 16 Befehlszyklen zwischen Interrupt-Start und EINT-Anweisung einzuhalten, um solche Effekte auszuschließen. Sie kann notfalls, wie im Beispielpogramm demonstriert, durch Einfügen eines oder mehrerer NOP-Befehle erreicht werden.

Für alle Anwendungen mit den A/D- und D/A-Wandlern ist zu beachten, daß dort ein Zahlenformat im Zweierkomplement mit negativem Vorzeichenbit verwendet wird. (0 entspricht dem größten negativen Wert, $2^{15}-1$ dem größten positiven.)

Da es sich um 14-Bit-Wandlerbausteine handelt, werden Bit 0 und Bit 1 im 16-Bit-Format als Null gelesen (Bit 0 des Wandlers liegt auf Bit 2 des Busses usw.).

Die fett-kursiv gedruckte Spalte des Hex-Programmcodes muß in das EPROM für die oberen 8 Bit (IC04), die andere Spalte entsprechend in EPROM IC03 geladen werden.

In dieses nur 21 Worte lange Testprogramm sind alle auf dem Signal-EMUF vorhandenen Schnittstellen und der Interrupt einbezogen worden. Wie man sieht, hält sich der Aufwand hierfür in Grenzen. Die Interrupt-Routine ohne Warteschleife ist übrigens 16 Befehlszyklen lang und weist damit eine Laufzeit von 3,2 µs auf.

Interessierte Anwender des Signal-EMUFs finden in verschiedenen Broschüren, die von Texas Instruments relativ preisgünstig angeboten werden, reichhaltige Sammlungen von Anwenderprogrammen. Dort sind Routinen für diverse Filter, Sprachverarbeitung, Modemübertragung sowie Regelungs- und Steuerungsprobleme abgedruckt (z. B. [3]...[8]).

Digitale Filter

Im Rahmen dieses Beitrages soll demonstriert werden, wie einfach digitale Filter mit dem 32010 realisiert werden können. Fast alle digitalen Filteralgorithmen basieren auf folgendem Zusammenhang zwischen dem Filtereingang $x_{(n)}$ und dem Filterausgang $y_{(n)}$:

$$y_{(n)} = \sum(a_k * y_{(n-k)}) + \sum(b_k * x_{(n-k)})$$

Dabei ist n der augenblickliche Abtastzeitpunkt, k die Zahl der berücksichtigten ver-

Der Befehlssatz des TMS 32010

Akkumulator-Befehle																			
Mnemo	Beschreibung	Zyklen	Worte	Opcode															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS	Absolutwert des Akkus	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0
ADD	Addiere um S nach links verschobenen Datenspeicher zum Akku	1	1	0	0	0	0	← S	→	1	←	←	←	←	←	D	←	←	←
ADDH	Addiere Datenspeicher zu den oberen 16 Bit des Akkus	1	1	0	1	1	0	0	0	0	0	1	←	←	←	D	←	←	←
ADDS	Addiere Datenspeicher zum Akku ohne Berücksichtigung des Vorzeichens	1	1	0	1	1	0	0	0	0	1	1	←	←	←	D	←	←	←
AND	AND-Verknüpfung Datenspeicher – Akku	1	1	0	1	1	1	1	0	0	1	1	←	←	←	D	←	←	←
LAC	Lade um S nach links verschobenen Datenspeicher in den Akku	1	1	0	0	1	0	← S	→	1	←	←	←	←	←	D	←	←	←
LACK	Lade Akku mit Konstante K (8 Bit)	1	1	0	1	1	1	1	1	1	0	←	←	←	←	K	←	←	←
OR	OR-Verknüpfung Datenspeicher – Akku	1	1	0	1	1	1	1	0	1	0	1	←	←	←	D	←	←	←
SACH	Lade die oberen 16 Bit des Akkus um X nach links verschoben in Datenspeicher	1	1	0	1	0	1	1	← X	→	1	←	←	←	←	D	←	←	←
SACL	Lade die unteren 16 Bit des Akkus in Datenspeicher	1	1	0	1	0	1	0	0	0	0	1	←	←	←	D	←	←	←
SUB	Subtrahiere um S nach links verschobenen Datenspeicher von Akku	1	1	0	0	0	1	← S	→	1	←	←	←	←	←	D	←	←	←
SUBC	Subtrahiere bedingt Datenspeicher von Akku (für Division)	1	1	0	1	1	0	0	1	0	0	1	←	←	←	D	←	←	←
SUBH	Subtrahiere Datenspeicher von oberen 16 Bit des Akkus	1	1	0	1	1	0	0	0	1	0	1	←	←	←	D	←	←	←
SUBS	Subtrahiere Datenspeicher ohne Berücksichtigung des Vorzeichens von Akku	1	1	0	1	1	0	0	0	1	1	1	←	←	←	D	←	←	←
XOR	XOR Verknüpfung Datenspeicher – Akku	1	1	0	1	1	1	1	0	0	0	1	←	←	←	D	←	←	←
ZAC	Akku = 0	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	1
ZALH	Akku = 0 und lade Datenspeicher in obere 16 Bit des Akkus	1	1	0	1	1	0	0	1	0	1	1	←	←	←	D	←	←	←
ZALS	Akku = 0 und lade Datenspeicher ohne Berücksichtigung des Vorzeichens in untere 16 Bit des Akkus	1	1	0	1	1	0	0	1	1	0	1	←	←	←	D	←	←	←

Ein-/Ausgabe- und Daten-RAM-Befehle																			
Mnemo	Beschreibung	Zyklen	Worte	Opcode															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMOV	Kopiert den Inhalt eines Datenspeichers in die darauffolgende Datenspeicheradresse	1	1	0	1	1	0	1	0	0	1	1	←	←	←	←	D	←	←
IN	Lädt Daten aus Port PA in Datenspeicher	2	1	0	1	0	0	0	← PA →	1	←	←	←	←	←	←	D	←	←
OUT	Lädt Daten aus Datenspeicher in PA	2	1	0	1	0	0	1	← PA →	1	←	←	←	←	←	←	D	←	←
TBLR	Lädt Daten aus dem Programmspeicher in den Datenspeicher	3	1	0	1	1	0	0	1	1	1	1	←	←	←	←	D	←	←
TBLW	Lädt Daten aus dem Datenspeicher in den Programmspeicher	3	1	0	1	1	1	1	1	0	1	1	←	←	←	←	D	←	←

GRUNDLAGEN

Verzweigungs-Befehle																				
Mnemo	Beschreibung	Zyklen	Worte	Opcode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	Verzweige ohne Bedingung	2	2	1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BANZ	Verzweige, wenn AR-Register nicht 0 ist	2	2	1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BGEZ	Verzweige, wenn Akku größer gleich 0	2	2	1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BGZ	Verzweige, wenn Akku größer 0	2	2	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BIOZ	Verzweige, wenn Anschluß BIO = 0	2	2	1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLEZ	Verzweige, wenn Akku kleiner gleich 0	2	2	1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BLZ	Verzweige, wenn Akku kleiner 0	2	2	1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BNZ	Verzweige, wenn Akku ungleich 0	2	2	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BV	Verzweige bei Überlauf	2	2	1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BZ	Verzweige, wenn Akku gleich 0	2	2	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CALA	Springe zu Unterprogramm Adresse in Akku	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0																
CALL	Springe zu Unterprogramm	2	2	1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RET	Rücksprung von von Unterprogramm	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1																

Befehle für die Register AR0, AR1 und DP																				
Mnemo	Beschreibung	Zyklen	Worte	Opcode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR	Lade AR-(R-)Register mit Datenspeicherzelle D	1	1	0 0 1 1 1 0 0 R 1																
LARK	Lade AR-(R-)Register mit Konstanten K	1	1	0 1 1 1 0 0 0 R																
LARP	Lade den Pointer ARP direkt mit K	1	1	0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0	K															
LDP	Lade den Seiten-Zeiger DP mit Datenspeicher	1	1	0 1 1 0 1 1 1 1 1																
LDPK	Lade den Seiten-Zeiger DP direkt mit K	1	1	0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0	K															
MAR	Modifiziert die AR-Register und den Pointer	1	1	0 1 1 0 1 0 0 0 1																
SAR	Speichere das AR-(R-)Register in Datenspeicher D	1	1	0 0 1 1 0 0 0 R 1																

Kontroll-Befehle																				
Mnemo	Beschreibung	Zyklen	Worte	Opcode	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINT	Verbiete Interrupt	1	1	0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0	1															
EINT	Erlaube Interrupt	1	1	0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0	1	0														
LST	Lade das Statusregister mit Datenspeicher	1	1	0 1 1 1 1 0 1 1 1																
NOP	Leerbefehl	1	1	0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0	0															
POP	Lade oberen Stackwert in den Akku	2	1	0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1	0	1														
PUSH	Lade oberen Stackwert aus dem Akku	2	1	0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1	0	0														
ROVM	Setze Überlaufmodus zurück	1	1	0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0	1	0														
SOVM	Setze Überlaufmodus	1	1	0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0	1	1														
SST	Speichere Statusregister in Datenspeicher	1	1	0 1 1 1 1 1 0 0 1																

gangenen Abtastzeitpunkte (wieviele berücksichtigt werden, hängt von der Ordnung des Filters ab) und a sowie b sind Filterkoeffizienten. Es handelt sich hier [9] um eine Differenzengleichung mit linearen konstanten Koeffizienten.

Man unterscheidet sogenannte FIR- (= Finite Impulse Response, d. h. nichtrekursive) und IIR- (= Infinite Impulse Response, also rekursive) Digitalfilter. Bei FIR-Filtern ist jeder Ausgangswert eine Funktion des gegenwärtigen und der vergangenen Eingangswerte. IIR-Filter berücksichtigen zusätzlich die vergangenen Ausgangswerte. FIR- und IIR-Filter unterscheiden sich in ihrem Filterverhalten und in ihrer Stabilität. Auf Details sowie die mathematische Behandlung dieser Problematik (Z-Transformation) soll an dieser Stelle nicht eingegangen werden (→ [10]).

Für FIR-Filter vereinfacht sich die oben angegebene Gleichung zu

$$Y(n) = \sum (b_k * x_{(n-k)}),$$

da alle $a_k = 0$. Konkret bedeutet das für ein FIR-Filter der Länge fünf, daß fünf Koeffizienten erforderlich sind, mit denen alle Eigenschaften des Filters eingestellt werden:

$$Y(n) = b_0 * x(n) + b_1 * x_{(n-1)} + b_2 * x_{(n-2)} + b_3 * x_{(n-3)} + b_4 * x_{(n-4)}$$

Die Koeffizienten b_k müssen Tabellenwerken [11] entnommen oder berechnet werden [9], [10]. Das entsprechende Programm für den Signal-EMUF steht in Bild 2. Dieses Beispiel ist als Interrupt-Routine aufgebaut. Einmalig aufzurufende Routinen zur Initialisierung des Zählers, laden der Register $b_{(0)} \dots b_{(4)}$ mit den erforderlichen Koeffizienten usw. müssen noch hinzugefügt werden. Diese doch recht aufwendige Berechnung mit immerhin 5 Integermultiplikationen erfordert lediglich 16 Programmzellen mit 18 Befehlszyklen (entspricht 3,6 µs).

Solch kleinere Programme für den TMS 32010 können durchaus „von Hand“ übersetzt werden, für größere Programme sollte man jedoch über einen Assembler verfügen. Das von Texas Instruments angebotene Entwicklungspaket ist recht gut, kostet aber leider 2000 DM. Ein einfacher Assembler für den TMS 32010 in Turbo-Pascal wird für 120 DM vom Autor dieses Beitrages angeboten.

In einer der nächsten Ausgaben von mc wird ein umfangreiches Beispielprogramm für ein einstellbares Bandpaßfilter auf dem

Ergänzungen zum Signal- EMUF

Beim täglichen praktischen Einsatz des Signal-EMUF haben sich noch ein paar kleine Korrekturen in der Dimensionierung verschiedener Bauteile ergeben, die die Betriebssicherheit dieses Rechners deutlich erhöhen:

IC01 Zwischen Pin 10 und 30 sollte ein keramischer Kondensator mit 0,1...0,47 µF gelötet werden

IC02 Es sollte ein CMOS-Quarzoszillator eingesetzt werden

R02 4,7 kΩ

R03 9,1 kΩ

R12 18 kΩ

Besonders für Anwendungen des Signal-EMUF in der Audio-Technik werden sehr hohe Ansprüche an die Qualität der verwendeten Filter gestellt. Die Ein- und Ausgangsfilter des Signal-EMUF stellen einen Kompromiß zwischen Kosten und Baugröße dar. Die Filter-Eckfrequenz ist getreu dem Abtasttheorem auf die Hälfte der Abtastfrequenz der Wandler eingestellt. Das Abtasttheorem gilt aber in dieser Form nur für ideale Filter, die real nicht existieren. In realen Systemen ist es daher sinnvoll, das Verhältnis nicht 1:2, sondern 1:>2 zu wählen. Da die Ansteuerfrequenz der Filterbausteine symmetrisch sein muß, wäre ein solches Verhältnis in der vorliegenden Schaltung jedoch nur mit einem eigenen programmierbaren Zähler für die Filteransteuerfrequenz realisierbar gewesen, was den Aufwand der Schaltung erheblich erhöht hätte.

Für die Praxis bedeutet die eingeschränkte Leistungsfähigkeit der Filter, daß u. U. unerwünschte Oberwellenanteile im Frequenzspektrum des Ausgangssignales nicht ganz unterdrückt werden, was man auch als erhöhten Klirrfaktor bezeichnen kann. Sollte sich dies in der Anwendung störend auswirken, so müssen anstelle der vorgesehenen höherwertige Filter eingesetzt werden. Aus Kostengründen sind dann aber Filter mit fest eingestellter Eckfrequenz zu empfehlen.

Harald Gosebruch

Befehle für Register T und P und für Multiplikation

Mnemo	Beschreibung	Zyklen	Worte	Opcode
				15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
APAC	Addiere P-Register zum Akku	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
LT	Lade das T-Register mit Datenspeicher	1	1	0 1 1 0 1 0 1 0 1 ← D →
LTA	LT und APAC in einem Befehl	1	1	0 1 1 0 1 1 0 0 1 ← D →
LTD	LT, APAC und DMOV in einem Befehl	1	1	0 1 1 0 1 1 0 0 1 ← D →
MPY	Multipliziere Akku mit T-Register und speichere Ergebnis in P-Register	1	1	0 1 1 0 1 1 0 1 1 ← D →
MPYK	Multipliziere T-Register mit Konstanten K und speichere Ergebnis mit P-Register	1	1	1 0 0 ← K →
PAC	Lade Akku mit P-Register	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0
SPAC	Subtrahiere P-Register vom Akku	1	1	0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0

Befehl	Kommentar
Start IN XN,3	Lade A/D Wandler-Wert in Register mit Namen
ZAC	Akkumulator = 0
LT XNM4 MPY B4	$x_{(n-4)} \cdot b_{(4)}$
LTD XNM3 MPY B3	$x_{(n-4)} \cdot b_{(4)} + x_{(n-3)} \cdot b_{(3)}$
LTD XNM2 MPY B2	$x_{(n-4)} \cdot b_{(4)} + x_{(n-3)} \cdot b_{(3)} + x_{(n-2)} \cdot b_{(2)}$
LTD XNM1 MPY B1	$x_{(n-4)} \cdot b_{(4)} + x_{(n-3)} \cdot b_{(3)} + x_{(n-2)} \cdot b_{(2)} + x_{(n-1)} \cdot b_{(1)}$
LTD XN MPY B0	$x_{(n-4)} \cdot b_{(4)} + x_{(n-3)} \cdot b_{(3)} + x_{(n-2)} \cdot b_{(2)} + x_{(n-1)} \cdot b_{(1)} + x_{(n)} \cdot b_{(0)}$
APAC	Addiere das Ergebnis der letzten Multiplikation zum Akkumulator
SACH YN,1	Speichere das Resultat in Register mit Namen YN
OUT YN,3	Ausgabe an den D/A-Wandler
EINT	Interruptfreigabe
WART B WART	Warteschleife für Interrupt

Bild 2. Diese Interrupt-Routine für das FIR-Filter mit fünf Koeffizienten ist in 3,6 µs abgearbeitet

Signal-EMUF vorgestellt werden. Digitale Filteralgorithmen sind, wie das Beispiel gezeigt hat, auf dem TMS 32010 recht einfach zu realisieren. Ein Problem ist die Einstellung der erforderlichen Koeffizienten. Das dann vorgestellte Programm wird sich diese Werte aus vorgegebener Mittenfrequenz, Bandbreite und Filterordnung selbst errechnen.

Literatur

- [1] Texas Instruments: Digital-Signalprozessor TMS 32010, Firmenschrift
- [2] Texas Instruments: TMS 32010 User's Guide, Firmenschrift
- [3] Texas Instruments: Digital Signal Processing Applications with the TMS 320 Family (SPRA012), Firmenschrift

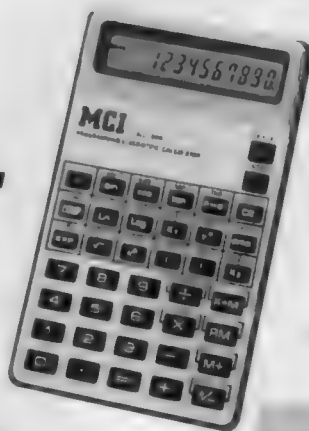
- [4] Texas Instruments: Implementation of FIR/IIR Filters with the TMS 32010, Firmenschrift
- [5] Texas Instruments: Companding Routines for the TMS 32010 (SPRA001), Firmenschrift
- [6] Texas Instruments: Precision Digital Sine-Wave Generation with the TMS 32010 (SPRA007), Firmenschrift
- [7] Texas Instruments: Matrix Multiplikation with the TMS 32010 (SPRA008), Firmenschrift
- [8] Texas Instruments: Control System Compensation and Implementation with the TMS 32010 (SPRA009), Firmenschrift
- [9] Oppenheim, Schläfer: Digital Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1975
- [10] Norbert Hesselmann: Digitale Signalverarbeitung, Vogel-Verlag, Würzburg, 1983
- [11] Lacroix, Witte: Zeitdiskrete normierte Tiefpässe, Hüthig Verlag, Heidelberg, 1980

... KOMPLETTPREISE...SYSTEMPAKETE

System Pakete für kluge Rechner ab 1349,-

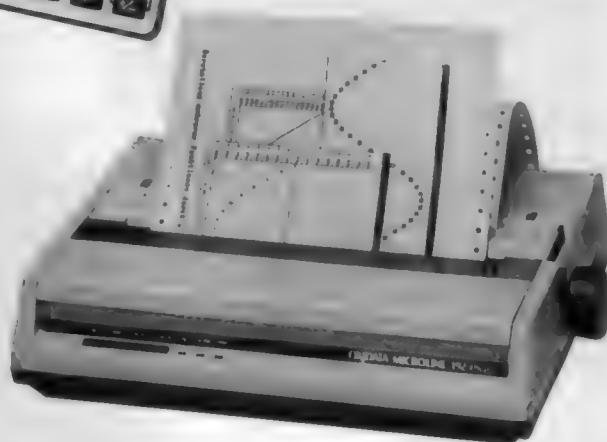


+



(+)

OKI ML192 Elite 899,-



System Paket 11

- MCI XT16SLC, 640 K, 1 x 360 K, Clock, ser. par., 12" Monitor, Tastatur
- MCI Programmierbarer Taschenrechner opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

10 MHz

1349,-

System Paket 21

- MCI XT16SLC, 640 K, 1 x 360 K, Clock, ser. par., 12" Monitor, Tastatur
- MCI 20MB Festplatte m. System formatiert
- MCI Programmierbarer Taschenrechner opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

10 MHz

1899,-

System Paket 31

- MCI AT4SLC, 640 K, 1 x 1, 2 MB, Clock, ser. par., 12" Monitor, Tastatur
- MCI Programmierbarer Taschenrechner opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

12 MHz

2399,-

System Paket 41

- MCI AT4SLC, 640 K, 1 x 1, 2 MB, Clock, ser. par., 12" Monitor, Tastatur
- MCI 20MB Festplatte m. System formatiert
- MCI Programmierbarer Taschenrechner opt. MCI 120 Zeichen Printer oder OKI ML 192 Elite (Mehrpreis siehe oben)

12 MHz

3199,-

KOMPATIBEL... 24-STUNDEN-TEST... LEISTUNG... PREIS...

PREIS... QUALITÄT... 1 JAHR GARANTIE

MCI XT 16 SLC

Grundausstattung ohne Monitor

ab **899,-**
beinhaltet:

- voll IBM® XT kompatibel
- 8088 CPU + 8087 Sockel
- 8 XT Slots
- 256 KB freier Speicher
- 1 x 360 KB Floppy-Disk
- Color- oder Monochr. Grafikkarte (Hercules II komp. 720 x 348 P.)
- Deutsche Normtastatur MK 5111
- 150 W Schaltnetzteil
- Parallele Drucker-Schnittstelle

Erweiterungen für XT 16 SLC-Serie

- 2. Laufwerk 360 KB **240,-**
- Speichererweiterung auf 640 KByte **a. A.**
- Clock/Seriell-Karte **70,-**
- I/O Plus II Karte **140,-**
- 20 MB Festplatte mit XT-Controller **+ 590,-**
- 30 MB Festplatte m. RLL XT-Contr. **+ 670,-**
- EGA-Set statt monochr. Karte **+ 1140,-**
- Opt. Maus mit Tablet **110,-**
- MS-DOS 3.3 deutsch + GW-Basic **+ 190,-**
- Professional Multifunktions-Tastatur MK 6000 **+ 100,-**
- 9" TTL Monitor grün **+ 150,-**
- 12" Monitor grün od. bern. **+ 220,-**
- 14" TTL Monitor grün, bern. od. weiß **+ 270,-**

Dieses Gerät ist nach den Bestimmungen d. VgV 1040/84 der Deutschen Bundespost funktionsfähig

NEU 4.7/10 MHz



MCI AT 4 SLC

Grundausstattung ohne Monitor

ab **1899,-**
beinhaltet:

- voll IBM® AT kompatibel
- 80286 CPU + 80287 Sockel
- 6 AT + 2 XT Slots
- 8 und 12 MHz umschaltbar
- 512 KB freier Speicher
- 1 x 1,2 MB/360 KB Laufwerk
- Color- oder Monochr. Grafikkarte (Hercules II komp. 720 x 348 P.)
- Parallele Drucker-Schnittstelle
- Batteriegep. Echtzeituhr/Kalender
- Kapazitive deutsche Normtastatur

Erweiterungen für AT 4 SLC-Serie

- 2. Laufwerk 360 KB **290,-**
- 20 MB Festplatte mit AT-Controller **880,-**
- 30 MB Festplatte m. RLL AT-Contr. **+ 990,-**
- Seriell-Karte **70,-**
- I/O Plus II Karte **140,-**
- EGA-Set statt monochr. Karte **+ 1140,-**
- MS-DOS 3.3 deutsch + GW-Basic **+ 190,-**
- Professional Multifunktions-Tastatur MK 6000 **+ 100,-**
- 9" TTL Monitor grün **+ 150,-**
- 12" Monitor grün od. bern. **+ 220,-**
- 14" TTL Monitor grün, bern./weiß **+ 270,-**

Dieses Gerät ist nach den Bestimmungen d. VgV 1040/84 der Deutschen Bundespost funktionsfähig

NEU 8/12 MHz



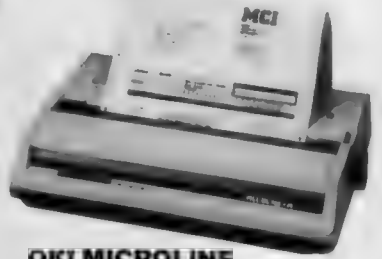
PRINTER



MCI Personal Computer Graphics Printer Plus

- voll kompatibel zum IBM Personal Computer Graphics Printer
- 120 Zeichen/sec.

399,-



OKI MICROLINE ML 192 Elite

- 9 Nadel Matrixdrucker
- Druckgeschwindigkeit 200 Z./sec.
- 40 Zeichen/sec. NLQ
- Druckpuffer 8 KB
- IBM Kompatibel

899,-

EGA



Hochauflösender EGA-Monitor

- Auflösung 320 x 200 (CGA Mode) 640 x 350 (EGA Mode)

EGA-Karte **249,-**

999,-

TELEFON-HOTLINE-PREISE MCI Telefonansage (20 sec.)

- (0 22 02)
- Festplatten & Controller **1081 40**
- Grafikkarten **1081 41**
- Monitore **1081 42**
- Schnittstellenkarten **1081 43**
- Drucker **1081 44**
- Aktuelle Neuigkeiten **1081 45**

12 Monate Garantie auf alle Geräte. Nach der Pang. Vo. v. 14.3.85 und wir bei Angeboten gegenüber dem Endverbraucher zur Angabe der Preise imi, liend, verpflichtet. Für Druckfehler wird nicht haftet. Preise gültig ab 1.7.88. Lieferzeit und Lieferbedingungen auf Anfrage. Änderungen der technischen Verbesserungen dienen vorbehalten. Zweck: Fernverkauf v. Einzelteilen. MCI MICRO COMPUTER INSTRUMENTS GMBH eingetragen AG. Bergisch Gladbach - HRB 2575. Herstellung und Vertrieb von Microcomputern. 5080 Bergisch Gladbach 2 - Bensberger Straße 252



**5060 Bergisch Gladbach 2
Bensberger Straße 252
Tel.-Nr.: 02202/1080
Fax: 02202/31009 · Telex: 8873518**

Uwe Labs

Hercules-Grafik-Bibliothek

Eine Grafik-Library in Turbo-Pascal für die Herculeskarte

Damit sich die Bibliothek ein wenig GSX/GKS-ähnlich strukturiert darstellt, wurden viele PASCAL-Bezeichnungen aus der Pascal-Schnittstelle von [1] übernommen. Weitere Ideen stammen aus dem Buch von Purgathofer [2], aus einigen Beiträgen der mc und der NEC-7220 Produktbeschreibung [3].

TGLIB ist eine Ansammlung von Include-Modulen in strenger Pascal-Hierarchie. Die nachfolgend besprochenen Teile umfassen drei Deklarations-Module und drei Unterprogramm-Module. Diese werden ins eigene Benutzerprogramm wie im ersten Programm (Bild 1) eingebunden. Bild 2 zeigt die Hardcopy der Grafik. Ein größeres Programm, das bereits viele der im Folgenden beschriebenen Grafik-Funktionen benutzt, ist das abgedruckte Programm FILLDEMO.PAS (Bild 3). Das Ergebnis in Bild 4 zeigt, daß auch komplizierte Figuren einwandfrei gefüllt werden.

Doch nun zur Einzelbeschreibung der TGLIB-Teile. Zu beachten ist, daß es zwei

Viele Anwender benötigen eine Grafik-Bibliothek für die Herculeskarte. mc stellt TGLIB vor, eine Grafik-Bibliothek, die in einer Hochsprache – dem weitverbreiteten Turbo-Pascal – programmiert ist. Sie ist ähnlich dem bekannten Grafikstandard GSX/GKS aufgebaut.

Sorten von Bezeichnern und Unterprogrammen gibt. Solche, die nur intern verwendet werden sollten und solche, die ausdrücklich für den Benutzer da sind (Tabelle 1). Im Gegensatz zu Modula-2 lassen sich die internen Teile in Pascal leider nicht verstecken.

Deklarationsmodule

TGLIBCON.INC (Bild 5) enthält Konstanten für die Herculeskarte, die Zeichenmodi, Linientypen und Farben. (Farben gibt's für die Herculeskarte natürlich nur 2, aber die TGLIB ist ja ursprünglich für den NEC 7220

– einem Grafikprozessor – geschrieben.)

Die Bildschirmabmessung ist geräteabhängig in Pixelanzahlen für Breite und Höhe angegeben, für die Herculeskarte also 720 x 348. Der Benutzer der TGLIB darf diese Größen jedoch nicht verwenden, sondern muß im Hinblick auf spätere Anpassungen an andere Grafikkarten die „virtuellen“ und oft anzutreffenden Abmessungen 1024 x 768 benutzen. Programme müssen dann für andere Grafikkarten nicht geändert werden.

TGLIBTYP.INC (Bild 6) enthält einige Typen für den Benutzer, einen Punkte-Array für Polygone (kann man auch selber deklarieren) und die hier nicht weiter beschrie-

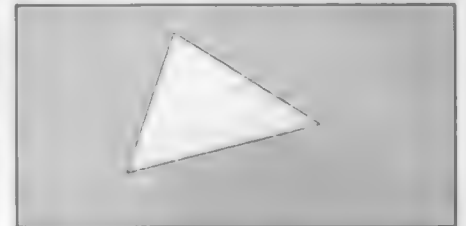


Bild 2. Die Ausgabe des Programms von Bild 1 als Hardcopy

```
PROGRAM Beispiell;

CONST
  (*$Ia:tglibcon.inc *)
TYPE
  (*$Ia:tglibtyp.inc *)

VAR
  (*$Ia:tglibvar.inc *)
  p : ARRAY[1..4,1..2] OF INTEGER;
  ch : CHAR;

  (*$Ia:tinit.inc*)
  (*$Ia:tlinef.inc*)

BEGIN
  openwo;          [ open workstation ]

  [ Ein Punkt: ]
  Dot(200,200);

  [ Ein Dreieck: ]
  p[1,1]:=100; p[1,2]:=100;
  p[2,1]:=500; p[2,2]:=200;
  p[3,1]:=200; p[3,2]:=400;
  p[4,1]:=p[1,1]; p[4,2]:=p[1,2];
  pline(p,4);      [ Zeichne Polygon ]

  read(kbd,ch);    [ Warte, bis Taste gedrückt wird ]
  closewo;         [ close workstation ]

END.
```

Bild 1. Zeichnen eines Dreiecks mit TGLIB

benen Kreis- und Ellipsentypen.

TGLIBVAR.INC (Bild 7) schließt die Deklarationsmodule mit einigen internen Variablen ab, von denen die Variable skala besonders wichtig ist, da sie die aktuelle Fenstergröße enthält.

Alle drei Modulen und der folgende müssen in einem TGLIB-Grafikprogramm genannt sein.

Das Basis-Modul TINIT.INC

Da dieses Modul die Initialisierung der Grafikkarte enthält, wurde als Name TINIT.INC gewählt. Inzwischen bietet es aber einiges mehr, so daß eigentlich der Name geändert werden müßte. Da dies aber die Änderung von vielen eigenen Programmen bedeuten würde, bleibt's halt dabei.

Eine grundlegende Entscheidung beim Entwurf der TGLIB war die Wortbreite von 16 Bit, die vom NEC 7220 übernommen wur-

```

PROGRAM FillTest; { Herkuleskarte, skaliert }
CONST
  (*$Ia:tglibcon.inc *)
  TYPE
    (*$Ia:tglibtyp.inc *)
  VAR
    (*$Ia:tglibvar.inc *)
    p : ARRAY[1..10,1..2] OF INTEGER;
    c : circ_type;
    ch : CHAR;
    i : INTEGER;
    (*$Ia:tinit.inc*)
    (*$Ia:tlines.inc*)
    (*$Ia:tskala.inc*)
  BEGIN
    openwo;
    gmode(tranmode);
    frame;
    window(50,50,700,700); frame;
    scale(0.0,10.0,FALSE,0.0,10.0,FALSE);

    { Ein Polygon }
    p[1,1]:=scx(3.9); p[1,2]:=scy(-1.0);
    p[2,1]:=scx(10.8); p[2,2]:=scy(4.2);
    p[3,1]:=scx(6.8); p[3,2]:=scy(6.5);
    p[4,1]:=scx(8.3); p[4,2]:=scy(8.3);
    p[5,1]:=scx(0.7); p[5,2]:=scy(5.2);
    p[6,1]:=scx(1.5); p[6,2]:=scy(2.0);
    p[7,1]:=scx(2.7); p[7,2]:=scy(5.2);
    p[8,1]:=scx(3.9); p[8,2]:=scy(0.0);
    fillstyle(solidfl);
    pline(p,8);

    { Ein Dreieck in xor-Modus }
    gmode(xormode);
    fillstyle(solidfl);
    p[1,1]:=scx(3.9); p[1,2]:=scy(0.7);
    p[2,1]:=scx(7.8); p[2,2]:=scy(1.3);
    p[3,1]:=scx(5.9); p[3,2]:=scy(9.8);
    p[4,1]:=scx(3.9); p[4,2]:=scy(0.7);
    pline(p,4);

    { Diverse Rechtecke, teils aus dem Window lappend }
    Rechteck(scx(2.0),scy(2.0),scx(6.0),scy(6.0),0);
    fillstyle(hollowfl);
    Rechteck(scx(5.5),scy(-1.0),scx(6.0),scy(6.0),0);
    fillstyle(solidfl);
    Rechteck(scx(-1.0),scy(5.0),scx(6.0),scy(6.0),0);

    { Neues Window }
    gmode(tranmode);
    window(800,500,200,200); frame;
    scale(0.0,10.0,FALSE,0.0,10.0,FALSE);
  
```

```

    { ... mit Sechseck }
    fillstyle(solidfl);
    c.x:=scx(5.0); c.y:=scy(5.0); c.r:=scx(4.0);
    Neck(c,6,0);

    { Neues Window }
    gmode(tranmode);
    window(800,300,200,200); frame;
    scale(0.0,10.0,FALSE,0.0,10.0,FALSE);

    { ... mit Dreieck }
    fillstyle(solidfl);
    p[1,1]:=scx(2.0); p[1,2]:=scy(2.0);
    p[2,1]:=scx(8.0); p[2,2]:=scy(2.0);
    p[3,1]:=scx(5.0); p[3,2]:=scy(8.0);
    p[4,1]:=p[1,1]; p[4,2]:=p[1,2];
    pline(p,4);

    { Neues Window mit Rechteck, dessen Fuelllinien gepunktet sind }
    gmode(tranmode);
    window(800,50,200,200); frame;
    scale(0.0,10.0,FALSE,0.0,10.0,FALSE);
    lintyp(dotted);
    Rechteck(scx(2.0),scy(2.0),scx(6.0),scy(6.0),0);

    read(kbd,ch);
    closewo;
  END.
  
```

Bild 3. Programmbeispiel zum Flächenfüllen

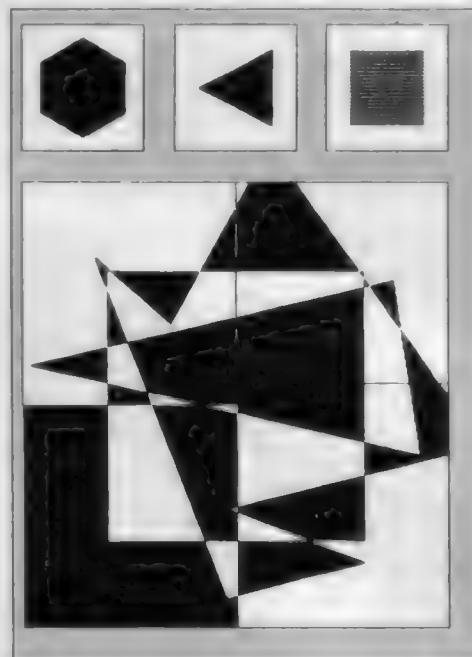


Bild 4. Hardcopy zum Programmbeispiel

Tabelle 1: Für den Benutzer zugängliche Bezeichner von TGLIB

Constanten aus TGLIBCON.INC: punkmax = 256 Punkte Maximum fuer Polyline vek_max = 60 Vektormaximum beim Polygonfüllen vek2_max = 120 Muß 2 * vek_max sein Zeichen - Modi: replmode Replace xormode XOR eramode Erase tranmode Transparent Fill - Stile: holowfil Holowfil = nur Linien solidfil Solidfil = Flächenfüllen Linien - Typen: solidlin Solid line = durchgezogene Linie dashlin Dash line = gestrichelte Linie dotlin Dot line = gepunktete Linie ddotlin Dash-Dot line = Strich-Punkt-Linie ldashlin Long dash line = langgestrichelte Linie Farben (fuer Hercules-Karte: schwarz=0, alles andere weiss): blackcol schwarz redcol rot greencol grün bluecol blau cyancol cyan (hellblau) yellowcol gelb magncol magentum (purpur) whitecol weiß Typen aus TGLIBTP.INC: circ1_type = RECORD Für Kreise y,x : INTEGER; r : INTEGER; END; ellipse_type = RECORD Für Ellipsen y,x : INTEGER; a,b : INTEGER; END; polyline_type = ARRAY [1..punkmax,1..2] OF INTEGER; Für Streckenzüge. Kann auch selbst bei Variablen kleiner definiert werden.		Aus TINIT.INC: PROCEDURE OpenWo Öffnet Workstation. Initialisiert den Graphikbildschirm PROCEDURE CloseWo Schließt Workstation. Schaltet wieder auf Text (Herculesk.) PROCEDURE Clear löscht gesamten Bildschirm PROCEDURE Gmode (p : INTEGER) Setzt Zeichenmodus. p = replmode, xormode, eramode oder tranmode PROCEDURE Dot (x,y : INTEGER) Setzt Punkt bei (x/y) Aus TLINEF.INC: PROCEDURE Lintyp (p : INTEGER) Setzt Linientyp. p = (siehe oben) PROCEDURE Lincol (p : INTEGER) Setzt Linienfarbe. p = (siehe oben) PROCEDURE Fillstyle (p : INTEGER) Setzt Füllstil. p = (siehe oben) PROCEDURE Fillpattern (p : INTEGER) Setzt Füllmuster. z.Zt nicht möglich. PROCEDURE Fillcol (p : INTEGER) Füllfarbe. p = (siehe oben) PROCEDURE Rotate (VAR pl; size,winkel,mx,my : INTEGER) Rotiert den Polyline-Array pl mit der Anzahl von size Punkten um den Winkel dw (links herum) um den Drehmittelpunkt mx,my. dw in 1/10-teil Grad: 0 <= dw <= 900. PROCEDURE PLine (var pl; size : INTEGER) Zeichnet die Poly-Line = Streckenzug im Array pl mit size Punkten. PROCEDURE Rechteck (x,y,laenge,breite,dw : INTEGER) Zeichnet Rechteck mit linker unterer Ecke (x/y). Drehwinkel dw wie oben. Wird um (x/y) gedreht. PROCEDURE Neck (p : circ1_type; n,dw : INTEGER) Zeichnet n-Eck um p.x/p.y als Mittelpunkt mit dem Radius p.r. Drehwinkel dw wie oben. Aus TSKALA.INC: PROCEDURE Window (x0,y0,b,h : INTEGER) Setzt ein Fenster im Bildschirm. (x0/y0) linke untere Ecke, b=Breite, h=Höhe. Beachte: x0*b < 1024 und y0*h < 768 PROCEDURE Frame Zeichnet einen Rahmen um das aktuellen Fenster mit den aktuellen Linienparametern. PROCEDURE Scale (wxmin,wxmax : REAL; wymin,wymax : REAL; lx : BOOLEAN; ly : BOOLEAN) Skaliert das aktuelle Fenster in Weltkoordinaten. lx/ly (nicht-) logarithmisch. (Logarithmisch ist noch nicht vollständig getestet!) FUNCTION scx (x : REAL) : INTEGER Rechnet die x-Welt- in x-Geräte-Koordinaten um FUNCTION scy (y : REAL) : INTEGER dito für die y-Werte
--	--	---

Tabelle 2: Interne Bezeichner von TINIT.INC.

EAD, DAD	Wort- und Pixeladresse im Videospeicher	transform	rechnet intern die 1024 x 768 Pixel in 720 x 348 um
graphic_mode	Ein/Ausschalten des Graphikmodus der Herculeskarte, Seite 1, wie schon öfters in MC beschrieben	back_transform	ist die dazu inverse Prozedur
init_vram	benutzt über Openwo die Prozedur graphic mode und initialisiert einige interne Variablen	in_window	prüft, ob ein Pixel im aktuellen Window liegt.
RMW	Read-Modify-Write = Ein 16-Bit-Wort wird aus dem Videospeicher gelesen, verändert und wieder zurückgeschrieben. Ein Verfahren vom NEC 7220.	L_Dot	ist eine interne Punktprozedur für den Linienalgorithmus. Damit die Adressberechnung schneller geht, wird nicht EAD benutzt, sondern die Berechnung an Ort und Stelle ausgeführt.

de. Dies wirkt sich mindestens auf waagerechte Linien (Flächenfüllen!) geschwindigkeitssteigernd gegenüber einer 8-Bit-Wortbreite aus.

Für den Benutzer stehen die Prozeduren Openwo, Closewo, Clear, Gmode und Dot zur Verfügung, deren Bedeutung der Tabelle 1 entnommen werden kann. Da die vorliegende TINIT für die Herculeskarte arbeitet, sind farbbestimmende Prozeduren herausgenommen. Tabelle 2 erläutert die internen Bezeichner von TINIT.

Der Linien- und Flächenfüll-Modul TLINEF.INC

Eine Benutzer-Prozedur LINE gibt es nicht. Es werden grundsätzlich nur Linienzüge gezeichnet. Eine einzelne Linie ist dann ein Linienzug mit nur je einem Anfangs- und einem Endpunkt. Schließt man einen Linienzug, so entsteht ein Polygon. Daher stammt der Name „pline“ der Prozedur, die Linienzüge zeichnet. Diese Struktur ermöglicht der TGLIB das Flächenfüllen von Polygonen.

Der Benutzer wählt sich eine Punkte-Array-Variable, z. B.:

p : ARRAY[1..4,1..2] OF INTEGER;

In diesen Array lassen sich nun die Koordinaten von Eckpunkten eines Linienzuges von hier drei Linien speichern. p[1,1] ist der x-Wert, p[1,2] der y-Wert des ersten Punktes, usw... (siehe Beispielprogramme in Bild 1 und Bild 3). Der Prozeduraufruf pline(p,4) zeichnet dann den Linienzug. pline(p,2) würde nur die erste, pline(p,3) nur die ersten zwei Linien zeichnen.


```

| Konstanten fuer TGLIB |

hindex= $3B4;      | 6845 Index-Register |
hdata = $3B5;      | 6845 Data-Register |
hmode = $3B8;      | Display Mode Control Port |
hstat = $3BA;      | Display Status Port |
hconf = $3BF;      | Configuration switch |

pageladr = $B800;   | Videospeicheradresse Page 1 |

wpz = 45;           | Worte pro Zeile |
xxmax = 719;        | Tatsaechliche Pixel pro Zeile - 1 (Gerateabhaengig) |
yymin = 347;        | Tatsaechliche Zeilenanzahl - 1 (Gerateabhaengig) |
vxmax = 1023;       | Gedachte Pixel pro Zeile - 1 |
vymax = 767;        | Gedachte Zeilenanzahl - 1 ( 3/4 des x-Wertes ) |
xxkor = 0.703;      | = (xxmax+1)/(vymax+1) |
yykor = 0.45;       | (= (vymax+1)/(vymax+1) Hiermit kann vorsichtig (!) |
                    | der Bildschirm korrigiert werden. Achte auf das |
                    | (= Zeichen !!! (bei Hercules-Karte (= 0.453125) |

punktmax = 256;     | Punkte Maximum fuer Polyline |
vek_max = 60;        | Vektormaximum beim Polygonfuellen |
vek2_max = 120;      | Muss 2 * vek_max sein |

| Es folgen die von grafcont.src uebernommenen Konstanten |

| Zeichen - Modi |
replmode = 1; xormode = 2; eramode = 3; tranmode = 4;

| Puell - Stile |
holowfl = 0; solidfl = 1;

| Linien - Typen |
solidlin = 1; dashlin = 2; dotlin = 3; ddotlin = 4; ldashlin = 5;

| Farben (fuer Hercules-Karte: schwarz=0, alles andere weiss) |
blackcol = 0; redcol = 1; greencol = 2; bluecol = 3; cyancol = 4;
yellocol = 5; magncol = 6; whitecol = 7;

```

Bild 5. Die Konstantendefinition von TGLIB

```

| (C) Labs. Teil der TGLIB (12. 9.87) : TGLIBTYP.INC |

| Typen fuer TGLIB |

circ_type = record
  y,x : integer;
  r : integer;
end;

ellipse_type = RECORD
  y,x : INTEGER;
  a,b : INTEGER;
END;

polyline_type = array [1..punktmax,1..2] of integer;

line_type = array [1..2,1..2] of integer; | nur intern ! |

grundfarbe = (BlauEbene,GruenEbene,RotEbene); | fuer NEC 7220 |

farbenen = ARRAY[BlauEbene..RotEbene] OF BOOLEAN;

pramregister = ARRAY[8..15] OF BYTE;

Skalarec = RECORD | Bezeichnungen nach Purgathofer |
  gxmin,gxmax, | Fensterkoordinaten. Ausschnitt der |
  gymin,gymax : INTEGER; | maximalen Geratekoordinaten |
  p,q,s,t : REAL; | Umrechnungswerte fuer Weltkoord. |
  logx,logy : BOOLEAN; | Schalter fuer log.Skalierung |
END;

```

Bild 6. Von TGLIB verwendete Typen

Vor dem Zeichnen von Linienzügen lassen sich noch mit den Prozeduren „Lintyp“, „Lincol“ und im Fall von Polygonen mit den Prozeduren „Fillstyle“, „Fillcol“ Typen und Farben bestimmen (siehe Tabelle). Die internen Prozeduren „line“, „hline“ und „fill“ übernehmen in Zusammenarbeit mit den Prozeduren „RMW“ und „L.Dot“ aus

TINIT die eigentliche Zeichenarbeit. Sowohl „line“ als auch „fill“ benutzen ein aus mc entnommenes Line-Clipping-Verfahren. Der Algorithmus der Prozedur line soll hier auch nicht mehr besprochen werden, da auch er aus mc stammt. Bevor ich den Flächenfüllalgorithmus bespreche, seien noch kurz die restlichen Pro-

zeduren von TLINEF erwähnt. „Rechteck“ zeichnet Rechtecke, „Neck“ zeichnet n-Ecke, die sich zudem noch drehen lassen. Dafür wird intern die Prozedur „Rotate“ verwendet, die aber auch dem Benutzer zur Verfügung steht. „Rotate“ dreht komplette Linienzüge.

Betrachten Sie für die folgende Beschreibung die Prozedur „fill“ im Listing von TLINEF.INC (Bild 9). Das Verfahren für das Flächenfüllen von Polygonen ist kurz in [2] beschrieben. Es werden Bildschirmlinie für Bildschirmlinie, den sogenannten Rasterlinien oder Scanlinien, die Schnittpunkte einer solchen Rasterlinie mit allen Polygonkanten berechnet. Zwischen diesen Schnittpunkten wird gegebenenfalls eine horizontale Linie mit Hilfe der schnellen Prozedur „hline“ gezeichnet.

Das Hauptproblem besteht darin, daß während des Flächenfüllens der Linienalgorithmus gleichzeitig für alle Polygonkanten ausgeführt werden muß. Laufen wir Rasterlinie für Rasterlinie vom tiefsten zum höchsten Punkt des Polygons, so muß bei jeder Rasterlinie reihum jede Polygonlinie (sofern sie von der Rasterlinie geschnitten wird) mit Hilfe des Linienalgorithmus um ein Pixel in y-Richtung fortschreitend bestimmt werden. Es findet auf die Polygonlinien bezogen sozusagen ein Multitasking statt. Deswegen habe ich für jede Linie des Polygons einen „Deskriptor“ eingeführt, der den jeweils momentanen Stand der Linienalgorithmusberechnung enthält. Da oben drein noch Sonderfälle für die Fülllinien auftreten (siehe unten) habe ich jede Linie des Polygons umgespeichert in einen „Vektor“, der außerdem immer nach oben gerichtet wird. Damit nicht zuviel Speicherplatz benötigt wird, ist über die Konstanten vek_max und vek2_max in TGLIBCON.INC die Eckenzahl des Polygons auf 60 festgesetzt, was aber geändert werden kann. In der Reihenfolge der Linien des Polygons in der Variablen p bzw. pl werden alle Linien zu Vektoren in die Variable sp umgespeichert und „positiv“ gerichtet.

Damit nicht alle Rasterlinien des gesamten Bildschirms in die Berechnung eingehen, wird nach der niedrigsten und der höchsten Rasterlinie gesucht (yrmin und yrmx), die das Polygon begrenzen. Um Doppelberechnungen zu verhindern, ist auch inzwischen durch Transformierung sichergestellt, daß nur mit geräteabhängigen Koordinaten und Rasterlinien gerechnet wird. Danach wird jedem Vektor ein Vektordescriptor in der Variablen v zugeordnet, der die für den Linienalgorithmus wichtigen Anfangsdaten für jeden Vektor (=Linie) enthält. In der mit „Füll-Routinen“ überschriebenen FOR-yr-Schleife wird jetzt für jede

Bild 7. Diese Variablen benötigt das Programmpaket

Bild 8. TINIT.INC ist der Kern von TGLIB

▶

```

port[hconf]=1;
port[hmode]=$28;
END;

END; (* graphic_mode *)

PROCEDURE init_vram;
BEGIN
  graphic_mode(TRUE); (* Graphik-Modus einschalten *)
  clear; (* löschen aller 3 Farbebenen *)
  (* Voreinstellungen *)
  p_linc:=1; p_linc:=1;
  p_text:=1; p_text:=1; p_text:=1; p_text:=2;
  p_filt:=0; p_filt:=1; p_filt:=1;
  p_mds:=1; p_mds:=1; p_mds:=1;
  p_mask:=ffff;
  p_polyline:=FALSE;
  END; (* init_vram *)

procedure gmode (p : integer);
begin
  case p of 2,3,4 : p_mds:=p else p_mds:=1 end
end; (* gmode *)

PROCEDURE sel_col (a : BYTE);
BEGIN
  (* nicht fuer Hercules-Karte *)
  END; (* sel_col *)

(* 2.Zt nicht benutzt:
FUNCTION in_absolutWindow (ax,ay : INTEGER) : BOOLEAN;
BEGIN
  WITH p_absolutSkala DO
    in_absolutWindow:=((axmin= ax) AND (ax<=axmax) AND
      (aymin= ay) AND (ay<=aymax))
  END; (* in_absolutWindow *)

PROCEDURE RMV (wortadre,neu : INTEGER);
(* Read-Modify-Write - Cycle *)
VAR
  wort : INTEGER;
BEGIN
  wort:=swap(newV[pageladr:wortadre]);

CASE p_mds OF
  replmode,
  tranmode : newV[pageladr:wortadre]:=swap(wort OR neu);
  xormode : newV[pageladr:wortadre]:=swap(wort XOR neu);
  erasmode : newV[pageladr:wortadre]:=swap(wort AND NOT(neu));
  ELSE newV[pageladr:wortadre]:=swap(wort OR neu);
END;
END; (* RMV *)

PROCEDURE change_col (p : INTEGER; VAR color : INTEGER);
BEGIN
  CASE p OF
    0..6 : color:=p ELSE color:=7
  END;
END; (* change_col *)

```

```

PROCEDURE change_zoom (p : INTEGER; VAR zoom : INTEGER);
BEGIN
  CASE p OF
    1..16 : zoom:=p ELSE zoom:=1
  END; (* change_zoom *)

procedure transform (var x,y : integer);
begin
  x:=trunc(x * xxkor); (* round ist besser als trunc, *)
  y:=trunc(y * yykor); (* aber 3 mal langsamer !!! *)
end; (* transform *)

PROCEDURE back_transform (VAR x,y : INTEGER);
BEGIN
  x:=trunc(x/xxkor)+1; (* round ist besser als trunc, *)
  y:=trunc(y/yykor)+1; (* aber 3 mal langsamer !!! *)
END; (* back_transform *)

procedure openwo;
begin
  init_vram;
  WITH skala DO
    BEGIN
      gxmin:=0; gxmax:=vxmax;
      gymin:=0; gymax:=vyymax;
      p:=1.0; q:=1.0;
      s:=0.0; t:=0.0;
      logx:=FALSE; logy:=FALSE;
      WITH p_absolutSkala DO
        BEGIN
          axmin:=gxmin; aymin:=gymin; transform(axmin,aymin);
          axmax:=gxmax; aymax:=gymax; transform(axmax,aymax);
        END;
      END;
    end; (* openwo *)

  PROCEDURE closewo;
  BEGIN
    graphic_mode(FALSE); (* Graphik-Modus ausschalten *)
  END; (* closewo *)

  FUNCTION in_window (gx,gy : INTEGER) : BOOLEAN;
  BEGIN
    WITH skala DO
      in_window:=((gxmin<=gx) AND (gx<=gxmax) AND
        (gymin<=gy) AND (gy<=gymax))
    END; (* in_window *)

  PROCEDURE L_Dot (x,y : INTEGER);
  (* Interne Linien-Dot-Prozedur. Da in der Linienprozedur ein
  Line-Clipping gemacht wird, braucht hier nicht mehr festgestellt
  werden, ob der Dot im Window liegt. *)
  VAR
    wortadre : INTEGER;
    wort : INTEGER;
  BEGIN
    (* Berechnung der Wortadresse geht hier schneller als neben
    die Funktionen EAD,dAD. SWAP ist wegen High/Low-Byte Vertauschung
    notwendig, bedeutet aber praktisch keinen Geschwindigkeitsverlust *)

```

```

( Y:=yyax-y; dx:= $2000*(y mod 4) + 96*(y div 4) + 2*(x div 16);
inline( $B/$Ymax/
$B/$6/y/
$29/$3/
$8B/$D3/
$81/$B3/$03/$04/
$B1/$D0/
$D3/$E3/
$D1/$E4/
$81/$E2/$FE/$FF/
$8B/$C2/
$D1/$E2/
$03/$C2/
$8B/$D6/
$B1/$04/
$D3/$E2/
$2B/$D6/
$03/$D4/
$8B/$96/$/
$B1/$04/
$D3/$E4/
$D1/$E2/
$03/$D4/
$89/$9E/$ortadresse);
mov dx,ax
add bx,dx
mov dx,[bp*x]; dx:=x
mov cl,4; CL:=4
shr dx,cl; dx:= x div 16
shl dx,1; dx:= 2*DX
add bx,dx; + 2*(x div 16)
mov [bp+y],bx
wort:=swap(memv[pageladr:wortadresse]);
CASE p_bods OF
replmode,
tranmode : memv[pageladr:wortadresse]:=swap((wort OR (1 shl (15 - (x mod 16)))) AND p_mask
);
k): xornode : memv[pageladr:wortadresse]:=swap((wort XOR (1 shl (15 - (x mod 16)))) AND p_mas
mask);
erarnode : memv[pageladr:wortadresse]:=swap((wort AND NOT(1 shl (15 - (x mod 16)))) AND p_
mask);
ELSE memv[pageladr:wortadresse]:=swap((wort OR (1 shl (15 - (x mod 16)))) AND p_mask);
END;
END; (* L_Dot *)
PROCEDURE Dot (x,y : INTEGER);
BEGIN
IF in_window(x,y)
THEN BEGIN
transform(x,y);
L_Dot(x,y);
END;
(* sel_col(p_linc); fuer Hercules-Karte nicht notwendig ! *)
END; (* L_Dot *)

```

```

( (C) Labs. Teil der TGLIB ( 28.10.87)
| MacOS - Version für HERCULES-Karte
|
| Objekt Code-Size + Datasize: 4848 Byte
|
PROCEDURE lincol (p : INTEGER);
BEGIN
CASE p OF
2..16: p_linc:=p ELSE p_linc:=1 END;
END; (* lincol *)
PROCEDURE lincol (p : INTEGER);
BEGIN
change_col(p,p_linc)
END; (* lincol *)
PROCEDURE lin_muster;
BEGIN
CASE p_linc OF
1: p_mask:=$fff; 2: p_mask:=$0ff; 3: p_mask:=$cccc;
4: p_mask:=$f0c; 5: p_mask:=$0fff;
ELSE p_mask:=$fff
END;
END; (* lin_muster *)
PROCEDURE fillstyle (p : INTEGER);
BEGIN
CASE p OF
1: p_filt:=1;
ELSE p_filt:=0;
END;
END; (* fillstyle *)
PROCEDURE fillpattern (p : INTEGER);
BEGIN
(* z.zt. sind keine Füllmuster zugelassen *)
END; (* fillpattern *)
PROCEDURE fillcol (p : INTEGER);
BEGIN
change_col(p,p_filt);
END; (* fillcol *)
PROCEDURE ClipPlane (VAR x1,y1,x2,y2 : INTEGER; VAR leer : BOOLEAN);
FUNCTION clip (x1,y1 : INTEGER; VAR x2,y2 : INTEGER) : BOOLEAN;
BEGIN
WITH stala DO
BEGIN
IF (y2 < gynin) AND (gynin <= y1)
THEN BEGIN
x2:=x1 + round((gynin-y1)/(y2-y1))*(x2-x1);
y2:=gynin;
ELSE IF (y1 <= gynax) AND (gynax < y2)
THEN BEGIN
x2:=x1 + round((gynax-y1)/(y2-y1))*(x2-x1);
y2:=gynax;
END;
END;
END;

```

Bild 9. Für das Zeichnen von Linien und Flächen wird diese Include-Datei gebraucht


```

IF (x2 < gxmin) AND (gxmin <= x1)
  THEN BEGIN
    y2:=y1 + round((gxmin-x1)/(x2-x1)*(y2-y1));
    x2:=gxmin;
  END
ELSE IF (x1 <= gxmax) AND (gxmax < x2)
  THEN BEGIN
    y2:=y1+round((gxmax-x1)/(x2-x1)*(y2-y1));
    x2:=gxmax;
  END;
clip:=(gxmin=x2) AND (x2<=gxmax) AND (gymin=y2) AND (y2<=gymax);
END;
END; (* clip *)

BEGIN
  leer:=NOT(clip(x1,y1,x2,y2) OR clip(x2,y2,x1,y1));
END; (* ClipLine *)

PROCEDURE line(p : line_type);
(* Siehe MC 11/86 Seite 12. Dieser Algorithmus ist schneller
als der Bresenham-Algorithmus. *)
VAR
  x,y,z,dx,dy,dz,i1,i2 : INTEGER;
  xmin,xmax,ymin,ymax : INTEGER;
  nicht_in_Fenster : BOOLEAN;
BEGIN
  xmin:=p[1,1]; ymax:=p[1,2]; xmax:=p[2,1]; ymax:=p[2,2];
  clipline(xmin,ymin,xmax,ymax,nicht_in_Fenster);
  IF nicht_in_Fenster THEN exit;
  transform(xmin,ymin); transform(xmax,ymax);
  sel_col(p_line); lin_muster;

  x:=abs(xmin-xmax); dy:=abs(ymin-ymax);
  IF xmin < xmax
  THEN BEGIN
    x:=xmin; y:=ymin;
    IF ymin > ymax THEN z:=1 ELSE z:=0;
  END
ELSE BEGIN
  x:=xmax; y:=ymax;
  IF ymax > ymin THEN z:=1 ELSE z:=0;
END;
IF dx > dy THEN i2:=dx ELSE i2:=dy;
dz:=i2 DIV 2;
IF p_polyline
THEN BEGIN
  IF nicht_in_Fenster
  THEN BEGIN
    L_Dot(x,y);
  END
ELSE
  THEN BEGIN L_Dot(x,y); i2:=i2-1; END;
END
ELSE L_Dot(x,y);
FOR i1:=1 TO i2 DO
  BEGIN
    IF dz < dx THEN BEGIN dz:=dx+dy; x:=x+1; END;
    IF dz >= dx THEN BEGIN dz:=dz-dx; y:=y+1; END;
    L_Dot(x,y);
  END;
END; (* line *)

```

PROCEDURE hline (x0,x1,y : INTEGER);
 (* Zeichnet horizontale Linie y zwischen x0 und x1.
 Diese Procedure führt kein Line-Clipping durch (!!!),
 da sie zum Flächenfüllen gedacht ist und das Line-Clipping
 dort durchgeführt wird. Es wird auch nicht geprüft, ob
 x0 < x1 ist, was unbedingt sein muß. *)
 VAR
 maske,w1,w2,w : INTEGER;
 BEGIN
 w1:=EAD(x0,y); (* Wortadresse Linien-Anfang *)
 w2:=EAD(x1,y); (* Wortadresse Linien-Ende *)
 sel_col(p_line); lin_muster;
 (* Falls nur ein Wort *)
 IF w1=w2
 THEN BEGIN
 maske:=(5ffff shr (x0 MOD 16)) AND (5ffff shl dad(x1)) AND p_mask;
 RMV(w1,maske);
 exit;
 END;
 (* Erstes Wort *)
 maske:=(5ffff shr (x0 MOD 16)) AND p_mask;
 RMV(w1,maske);
 (* Zweites bis vorletztes Wort *)
 maske:=5ffff AND p_mask;
 w:=w1+2;
 WHILE w <= w2-2 DO
 BEGIN
 RMV(w,maske);
 w:=w+2;
 END;
 (* Letztes Wort *)
 maske:=5ffff shl dad(x1) AND p_mask;
 RMV(w2,maske);
 END; (* hline *)

PROCEDURE fill (VAR pl; size : INTEGER);
 (* Polygonfüllalgorithmus mittels horizontaler Linien.
 Siehe auch "Graphische Datenverarbeitung (Purgathofer)".
 Dort stehen die grundlegenden Ideen.
 Linien-Algorithmus nach MC 11/86 Seite 12, abgewandelt fuer
 positiv gerichtete Vektoren. *)
 TYPE
 VektorDescriptor = RECORD
 x,y : INTEGER;
 xmax : INTEGER;
 dx : INTEGER;
 dy : INTEGER;
 dz : INTEGER;
 z : INTEGER;
 spalte : INTEGER;
 END;
 aktuelle x/y-Werte : VektorDescriptor;
 maximaler x-Vert : INTEGER;
 aus dem Linienalg. : VektorDescriptor;
 " : VektorDescriptor;
 " : VektorDescriptor;
 " : VektorDescriptor;
 aktuelle Bildschirmspalte : VektorDescriptor;

```

VektorPolygon = ARRAY[1..vek_max,1..2,1..2] OF INTEGER; { Vektor-Polygon }
1.Komponente : Vektornummer
2.Komponente : 1= Anfangspunkt
               2= Endpunkt
3.Komponente : 1= x-Wert
               2= y-Wert

VAR
p : polyline_type absolute p1;
sp : VektorPolygon;
nf : BOOLEAN;
v : ARRAY[1..vek_max] OF VektorDescriptor;
s : ARRAY[1..vek2_max] OF INTEGER; { Schnittpunkte in einer y-Scanline }
l,j : INTEGER;
k : INTEGER;
n : INTEGER;
sx : INTEGER;
ymin,ymax : INTEGER;
yr : INTEGER;
max,min : INTEGER;
xmin,ymin,ymax,i2 : INTEGER;

{ Wenn's zu viele Vektoren sind, dann mach nichts }
IF size > vek_max THEN exit;

{ p wird nach sp umgespeichert; die x,y-Werte werden transformiert. }
{ Ausserdem werden alle Vektoren positiv gerichtet. }
size:=size-1; { Anzahl der Vektoren }
k:=1;
FOR i:=1 TO size DO
BEGIN
sp[k,1,1]:=p[i,1]; sp[k,1,2]:=p[i,2];
sp[k,2,1]:=p[i+1,1]; sp[k,2,2]:=p[i+1,2];
transform(sp[k,1,1],sp[k,1,2]);
transform(sp[k,2,1],sp[k,2,2]);
IF sp[k,2,2] - sp[k,1,2] < 0
THEN BEGIN
{ wenn neg.Richtung, dann vertausche }
j:=sp[k,2,1]; sp[k,2,1]:=sp[k,1,1]; sp[k,1,1]:=j;
j:=sp[k,2,2]; sp[k,2,2]:=sp[k,1,2]; sp[k,1,2]:=j;
END;
k:=k+1;
END;

{ Suche die minimale und maximale Raster-Linie }
{ Berechne y-Randwerte (geratespezifisch) fuer vertikales Clipping.
min: unterer Rand, max: oberer Rand, k ist Dummy }
min:=skala.ymin; max:=skala.ymax; { noch gerate-unabhängig }
transform(k,min); transform(k,max); { jetzt geratespezifisch }
ymin:=min; ymax:=max; { Echte Raster-Linien-Anzahl }

FOR k:=1 TO size DO
FOR i:=1 TO 2 DO
BEGIN
IF sp[k,1,2] > ymax THEN ymax:=sp[k,1,2];
IF sp[k,1,2] < ymin THEN ymin:=sp[k,1,2];
END;
IF ymax < min THEN ymin:=min;
IF ymax > max THEN ymax:=max;
}

Herstellen der Vektor-Descriptors fuer den Liniensalgorithmus
FOR k:=1 TO size DO
WITH v[k] DO
BEGIN
xmin:=sp[k,1,1]; ymin:=sp[k,1,2];
xmax:=sp[k,2,1]; ymax:=sp[k,2,2];
dx:=abs(xmin-xmax); dy:=abs(ymin-ymax);
x:=xmin; y:=ymin;
IF xmin < xmax THEN z:=1 ELSE z:=-1;
IF dx > dy THEN i2:=dx ELSE i2:=dy;
dz:=i2 DIV 2;
spalte:=xmin;
END;

{ Fuell - Routinen: }
FOR yr:=ymin TO ymax DO
BEGIN
n:=0;
FOR k:=1 TO size DO
{ Falls Vektor k und Raster-Linie yr sich schneiden, dann ... }
{ Die Vektorspitzen werden weggelassen. Damit werden
zwei aufeinander stossende Spitzen 0-mal, aufeinander
stossende Anfange und Spitzen 1-mal beruecksichtigt.
Ausnahme: Die Spitzen liegen auf ymax und der Vektor
liegt nicht auf der Rasterlinie ymax }
IF ((sp[k,1,2]<yr) AND (yr < sp[k,2,2])) OR
((yr=ymax) AND (ymax=sp[k,2,2]) AND (ymax<sp[k,1,2]))
THEN BEGIN
{ berechne den Schnittpunkt }
WITH v[k] DO
BEGIN
{ merke den Schnitt von yr mit einem Rand der Linie k }
n:=n+1; s[n]:=x; sx:=x;
REPEAT
IF dz < dx THEN BEGIN dz:=dz+dy; x:=x+z; END;
IF dz > dx THEN BEGIN dz:=dz-dx; y:=y+1; END;
IF y=yr THEN sx:=x;
spalte:=spalte+z;
UNTIL (y > yr) OR (spalte=xmax);
{ merke Schnitt von yr mit dem anderen Rand der Linie k }
n:=n+1; s[n]:=sx;
END;
END;
}

{ Sortiere die Schnittpunkt aufsteigend nach x-Werten }
FOR i:=2 TO n DO
FOR k:=n DOWNTO i DO
IF s[k-1] > s[k]
THEN BEGIN
j:=s[k-1]; s[k-1]:=s[k]; s[k]:=j;
END;
END;

```

```

| Berechne x-Randwerte (geratespezifisch) fuer horizontales Clippen.
min:=linker Rand, max:=rechter Rand, k ist Duany |
min:=skala.gmin; max:=skala.gmax; | noch gerate-unabhängig |
transform(min,k); transform(max,k); | jetzt geratespezifisch |

| Zeichne zwischen je 2 Schnittpunkten die horizontale Verbindung |
k:=1;
WHILE k <= n DO
  BEGIN
    IF s[k] < min THEN s[k]:=min; | Clip links |
    IF s[k+1] > max THEN s[k+1]:=max; | Clip rechts |
    hline(s[k],s[k+1],yr); | schnelle horizontale Linie |
    k:=k+4;
  END;
END;
END; (* fill *)

PROCEDURE Rotate (VAR pl: size; winkel,ax,ay: INTEGER);
(* pl Polyline-Array mit der Anzahl von size Punkten.
Drehwinkel dw in 1/10-tel Grad: 0 <= dw <= 900.
ax,ay Drehmittelpunkt. *)
VAR
  w: REAL;
  i: INTEGER;
  x,y: REAL;
  p: polyline_type absolute pl;
  w:=winkel; w:=w/10; w:=pi*w/180;
  FOR i:=1 TO size DO
    BEGIN
      x:=p[i,1]-ax; y:=p[i,2]-ay;
      p[i,1]:=trunc(x*cos(w) + y*sin(w)) + ax;
      p[i,2]:=trunc(-x*sin(w) + y*cos(w)) + ay;
    END;
  END; (* Rotate *)

procedure pline (var pl: size: integer);
var
  p: polyline_type absolute pl;
  pt: line_type;
  i: integer;
  farbe_alt: INTEGER;
begin
  IF p_filt <= holowfl
  THEN BEGIN
    farbe_alt:=p_plinc;
    p_plinc:=p_filt;
    fill(p,size);
    p_plinc:=farbe_alt;
  END
  ELSE BEGIN
    i:=0; p_polyline:=TRUE;
    repeat
      i:=i+1;
      pt[i,1]:=p[i,1]; pt[i,2]:=p[i,2]; | i. Punkt |
      pt[i+1,1]:=p[i+1,1]; pt[i+1,2]:=p[i+1,2]; | i+1. Punkt |
      line(pt);
      until i=size-1;
      p_polyline:=FALSE;
    END;
  end; (* pline *)

```

```

PROCEDURE Rechteck (x,y,laenge,breite,dw: INTEGER);
(* Rechteck mit linker unterer Ecke (x/y).
Drehwinkel dw in 1/10-tel Grad: 0 <= dw <= 3600
Wird um (x/y) gedreht. *)
VAR
  p: ARRAY[1..5,1..2] OF INTEGER;
  BEGIN
    WITH skala DO
      BEGIN
        laenge:=round(laenge-s); | tatsächliche Laenge in x-Richtung |
        breite:=round(breite-t); | tatsächliche Laenge in y-Richtung |
      END;
      p[1,1]:=x; p[1,2]:=y;
      p[2,1]:=x+laenge; p[2,2]:=y;
      p[3,1]:=p[2,1]; p[3,2]:=y+breite;
      p[4,1]:=x; p[4,2]:=p[3,2];
      p[5,1]:=p[1,1]; p[5,2]:=p[1,2];
      IF dw<>0 THEN Rotate(p,5,dw,x,y);
      pline(p,5);
    END; (* Rechteck *)
  END;

PROCEDURE Reck (p: circ_type; n,dw: INTEGER); | regelmaessig |
CONST
  | Drehwinkel dw in 1/10-tel Grad: 0 <= dw <= 3600 |
  | m-1 = Maximum fuer die Eckenzahl n !!! |
  m:=51;
VAR
  x: INTEGER; | ab n=50 kreisfoermig |
  phi: REAL;
  z: REAL;
  rx,ry: REAL;
  pl: ARRAY[1..m,1..2] OF INTEGER;
  BEGIN
    rx:=p.r;
    WITH skala DO
      BEGIN
        rx:=rx-s; | Radius in x-Richtung, tatsächliche Laenge |
        ry:=round(q*rx/p); | Passe in y-Richtung an |
      END;
      phi:=2*pi/n;
      FOR x:=0 TO n DO
        BEGIN
          pl[x+1,1]:=trunc(rx*cos(x*phi))+p.x;
          pl[x+1,2]:=trunc(ry*sin(x*phi))+p.y;
        END;
      IF dw<>0 THEN Rotate(pl,n+1,dw,p.x,p.y);
      pline(pl,n+1);
    END; (* Reck *)
  END;

```

```

| (C) Labs. Teil der TGLIB (27.12.87)                                : TSKALA.INC |
| Objekt Code-Size + Datasize: 1072 Byte                               |
| Mit den folgenden Unterprogrammen lassen sich die Welt/Viewport-    |
| Koordinatenumrechnungen durchfuehren. Der Anwender rechnet mit     |
| REAL-Weltkoordinaten. |

PROCEDURE window (x0,y0,b,h : INTEGER); | Proc Viewport nach Purgathofer |
VAR                                       | definiert Zeichenfenster      |
  xmax,yamax : INTEGER;
BEGIN
  xmax:=vxkxax; ymax:=vyymax;
  WITH skala DO
    BEGIN
      IF (0<=x0) AND (x0<=xmax)
      THEN gxmin:=x0 ELSE gxmin:=0;
      IF (0<=y0) AND (y0<=ymax)
      THEN gymin:=y0 ELSE gymin:=0;
      IF (0<=x0+b) AND (x0+b<=xmax)
      THEN gxmax:=x0+b ELSE gxmax:=xmax;
      IF (0<=y0+h) AND (y0+h<=ymax)
      THEN gymax:=y0+h ELSE gymax:=ymax;
      WITH p_absolutSkala DO
        BEGIN
          axmin:=gxmin; aymin:=gymin; transform(axmin,aymin);
          axmax:=gxmax; aymax:=gymax; transform(axmax,aymax);
        END;
      END;
    END;
  END; (* window *)

PROCEDURE frame; | Rahmen mit den aktuellen Linienparametern ums Fenster |
VAR
  l : line_type;
BEGIN
  WITH skala DO
    BEGIN
      l[1,1]:=gxmin; l[1,2]:=gymin; l[2,1]:=gxmax; l[2,2]:=gymax; line(l);
      l[1,1]:=gxmax; l[1,2]:=gymax; l[2,1]:=gxmin; l[2,2]:=gymin; line(l);
      l[1,1]:=gxmin; l[1,2]:=gymax; l[2,1]:=gxmax; l[2,2]:=gymin; line(l);
      l[1,1]:=gxmax; l[1,2]:=gymin; l[2,1]:=gxmin; l[2,2]:=gymax; line(l);
    END;
  END; (* frame *)

PROCEDURE scale (wxmin,wxmax : REAL; lx : BOOLEAN; | nach Purgathofer: |
  wymin,wymax : REAL; ly : BOOLEAN); | Window --> Unrechnerwerte |
BEGIN
  WITH skala DO
    BEGIN
      logx:=lx; IF lx THEN BEGIN wxmin:=ln(wxmin); wxmax:=ln(wxmax) END;
      logy:=ly; IF ly THEN BEGIN wymin:=ln(wymin); wymax:=ln(wymax) END;
      p:=(gxmax-gxmin)/(wxmax-wxmin);
      q:=(gymax-gymin)/(wymax-wymin);
      s:=(gxmin*wxmax - gxmax*w xmin)/(wxmax-wxmin);
      t:=(gymin*wymax - gymax*wymin)/(wymax-wymin);
    END;
  END; (* scale *)

FUNCTION in_range (x,z,y : REAL) : BOOLEAN;
BEGIN
  in_range:=(x=z) AND (z=y);
END; (* in_range *)

FUNCTION scx (x : REAL) : INTEGER; | Welt --> Gerat fuer x-Werte |
BEGIN
  WITH skala DO
    BEGIN
      IF logx THEN x:=ln(x);
      scx:=round(p*x + s);
    END;
  END; (* scx *)

FUNCTION scy (y : REAL) : INTEGER; | Welt --> Gerat fuer y-Werte |
BEGIN
  WITH skala DO
    BEGIN
      IF logy THEN y:=ln(y);
      scy:=round(q*y + t);
    END;
  END; (* scy *)

```

Bild 11. Für die Skalierung der Zeichnung braucht man TSKALA.INC

einzelne Rasterlinie von der untersten Rasterlinie, die das Polygon eben noch schneidet bis zu obersten die Schnittpunkte aller Vektoren mit der aktuellen Rasterlinie berechnet. Dies geschieht in der FOR-Schleife. Hier wird mit Hilfe der Vektorkoordinaten jeder Vektor für die aktuelle Rasterlinie (falls er sie schneidet) in y-Richtung um ein Pixel weitergerechnet. Der Linienalgorithmus wird wie mit einer Zeitscheibe über alle Vektoren geschaltet. Die so innerhalb einer Rasterlinie gefundenen x-Werte der Schnittpunkte werden in der Variablen s gespeichert und danach aufsteigend nach x-Werten sortiert (Bubblesort, denn es sind ja nicht viele). Mit den TGLIB-Systemdaten wird dann ein gerätespezifisches horizontales Windowclipping vorbereitet, das in der folgenden WHILE-Schleife gegebenenfalls die Schnittpunkte, sollten sie außerhalb des aktuellen Windows liegen, korrigiert. In dieser WHILE-Schleife werden zwischen je zwei Schnittpunkten unter Zuhilfenahme der schnellen Prozedur hline die Fülllinien gezeichnet. Da fast immer 16 Pixels gleichzeitig gezeichnet werden, ist die Füllgeschwindigkeit recht gut. Beim Berechnen der Fülllinien ist mit der IF-Anweisung, die der FOR-Schleife folgt, auf die Sonderfälle, die in Bild 10 angegeben sind, Rücksicht genommen. Rasterlinie a ist sozusagen der Normalfall. Der Schnittpunkt der Rasterlinie b mit den Vektoren 2 und 3 darf keinmal oder zweimal berücksichtigt werden. Bei Rasterlinie c darf der Schnittpunkt, bei dem sich die Vektoren 1 und 5 berühren, nur einmal benutzt werden. Diese Sonderfälle werden ganz einfach dadurch gelöst, indem man die Vektorspitzen überhaupt nicht berücksichtigt. Bei b treten die Endpunkte von 2 und 3 nicht auf, d. h. es wird eine durchgehende statt zwei einzelnen Fülllinien gezeichnet. Bei c tritt nur der Anfangspunkt von 5 auf (der ja identisch mit dem Endpunkt von 1 ist).

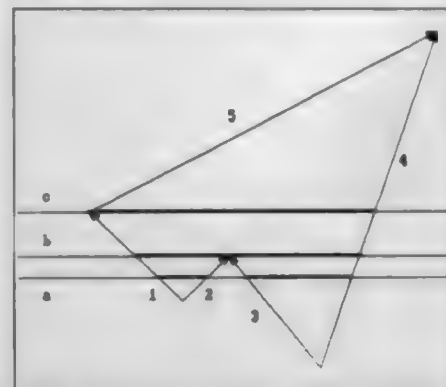


Bild 10. Beim Flächenfüllen muß man auf ein paar spezielle Probleme achten

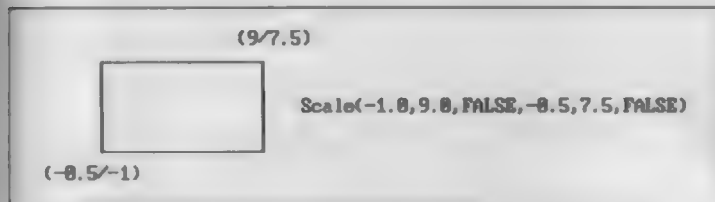


Bild 12. Skalierte Zeichenfläche

Skalierung mit TSKALA.INC

Einen wesentlichen Schritt zur hardware-unabhängigen Grafikprogrammierung macht der Benutzer, wenn er mittels einer Skalierung nicht in Gerätekoordinaten sondern in Weltkoordinaten programmiert. Weltkoordinaten sind der Anwendung angepaßt und deshalb vernünftiger als Gerätekoordinaten. Die dazu notwendigen Unterprogramme befinden sich im Modul TSKALA.INC (Bild 11). Mit Window wird ein Fenster aus dem Bereich 1024 x 768 deklariert. Z. B. ist durch Window (100, 200, 500, 400) ein Fenster mit dem linken unteren Eckpunkt (100/200),

der Breite 500 und der Höhe 400 festgelegt. Mit der Prozedur Frame kann noch ein Rahmen darum gezeichnet werden. Die eigentliche Skalierung erfolgt durch die Prozedur Scale, die in x-Richtung und in y-Richtung das Fenster in reellen(!) Weltkoordinaten skaliert. Das obige Fenster wäre mit Scale(-1.0,9.0,FALSE,-0.5,7.5,FALSE) nichtlogarithmisch so anzusehen, wie in Bild 12.

Es stellt wegen der Fenster-Anpassung einen Teil des wirklichen Bildschirm dar (siehe auch die Ausgabe des Programms FILLDEMO, das mehrere Fenster verwendet). Da die TGLIB aber natürlich intern nur ganzzahlige Koordinaten verarbeiten kann,

müssen alle Werte mit den Funktionen scx und scy umgerechnet werden (dies ist ebenfalls gut im Programm FILLDEMO zu sehen).

Will man jetzt die gesamte Grafik vergrößern oder verkleinern oder an anderer Position darstellen, so ist nur (!) der Window-Aufruf zu ändern. Dies ist auch ein großer Vorteil der Programmierung mit den der Anwendung angepassten Weltkoordinaten. Vielleicht gibt dieses TGLIB-Paket Ihnen Anregungen zur Grafikprogrammierung. Möglicherweise finden Sie Ideen zur Verbesserung und Geschwindigkeitssteigerung.

Literatur:

- [1] Pascal-Schnittstelle zu einer GLIB von Oettle & Reichler für den NEC 7220 Grafikprozessor
- [2] Graphische Datenverarbeitung, Werner Purgathofer. Springer Verlag Wien New York 1985
- [3] Product Description. Graphic Display Controller uPD 7220. 6/84 V3.2 NEC Electronics
- [4] mc-Beiträge aus 11/86, 2/87, 3/87, 4/87, 6/87, 7/87, 9/87, 10/87

Diskettenzugriff ohne Problem

Wer kennt nicht das Problem: Das Programm verlangt vom Benutzer eine Diskette in einem bestimmten Laufwerk, um eine Datei von dieser zu lesen oder neue Daten darauf zu schreiben. Wenn eine falsche Diskette im Laufwerk steckt, läßt sich der Fehler recht einfach abfangen. Ist der Benutzer aber zerstreut und vergißt er den Hebel am Laufwerk umzulegen, wird die Bildschirm-Maske von der DOS-Meldung „Gerät nicht bereit ...“ ruiniert. Es wurde schon früher eine Lösung des Problems über eine programmeigene Behandlung des dafür zuständigen Interrupts 24H vorgeschlagen. Wer aber mit der Programmierung von Interrupt-Handleinern noch nicht so bewandert ist, schreckt möglicherweise vor der Verwendung zurück.

Das Programm im Bild zeigt eine Lösung, die ohne Interrupt-Handler auskommt. Die Funktion „TestDrive“ erkundigt sich über den BIOS-Aufruf \$11 zunächst, ob das angesprochene Diskettenlaufwerk überhaupt existiert. Dabei gibt die Variable „Drive“ die (bei 0 beginnende) physikalische Zählung der Laufwerke an. Ist das Laufwerk vorhanden, wird über den BIOS-Disketteninterrupt \$13 ein Prüfließen versucht. Das ist natürlich nur dann erfolgreich, wenn eine Diskette eingelegt und die Laufwerks-

```

Program Drives_Vorhanden;

type
  Treg = record case Boolean of
    false : (ax,bx,cx,dx,bp,di,si,ds,es,flags : Integer);
    true  : (al,ah,bl,bh,cl,ch,dl,dh          : Byte);
  end;

Function TestDrive(Drive : Byte) : Boolean;
{ Testen, ob Laufwerk bereit.
  Zuordnung: 0: LW A, 1: LW B, usw. }
var
  Regs      : Treg;
  Num_Drives : Integer;
begin
  Intr($11,Regs); { Anzahl Laufwerke feststellen }
  Num_Drives := Regs.AX;
  if (Num_Drives and 1) = 0 then
    Num_Drives := 0 { Equipment: Bit 0=0 kein LW }
  else
    Num_Drives := ((Num_Drives shl 8) shr 14) + 1;
  if Num_Drives < (Drive + 1) then { Laufwerk gibt es nicht }
    TestDrive := false
  else
    with Regs do
    begin
      dl := drive; { Disk: 0 .. 3, HDD: $80..$81 }
      dh := 0; { Kopf 0 }
      cx := 1; { Spur 0, Sektor 1 }
      ax := $0401; { Verify one Sector }
      Intr($13,Regs); { Disketteninterrupt }
      TestDrive := ah<>$80; { $80:Timeout/Drive not Ready }
      dl := drive; { Disk-Reset }
      ax := 0; { sonst Fehler beim folgenden }
      Intr($13,Regs); { Zugriff! }
    end;
  end;

begin
  if TestDrive(0) then Writeln('A: ready') else Writeln('A: not ready');
  if TestDrive(1) then Writeln('B: ready') else Writeln('B: not ready');
  if TestDrive(2) then Writeln('C: ready') else Writeln('C: not ready');
end.

```

Zuverlässiger Zugriff auf Disketten

klappe geschlossen ist. Das Testprogramm versucht die beiden existierenden Laufwerke A und B sowie das nicht vorhandene dritte Laufwerk anzusprechen.

Die Funktion ist auch für Messedemos

nützlich, die auf Diskette ablaufen. Ruft man „TestDrive“ in regelmäßigen Abständen auf, kann der Computer sofort geräuschvoll Alarm schlagen, wenn jemand versucht, die Diskette zu entfernen.

Jürgen Plate

Michael Bauer

Maus-Funktionen für Turbo-Basic

Turbo-Basic bietet mächtige Möglichkeiten zur Definition von Funktionen und Prozeduren, daher können die fehlenden Prozeduren in einer Bibliothek definiert und bei Bedarf mit der Compiler-Anweisung `$INCLUDE` in den Quelltext eingebunden werden.

In *Bild 1* (MAUSFUNC.BIB)

sind alle für den Betrieb einer Microsoft- bzw. dazu kompatiblen Maus benötigten Routinen aufgenommen worden. Die Prozeduren halten sich an die Angaben des 'Microsoft Mouse Programmer's Reference

Der Befehlsumfang von Turbo-Basic ist erfreulich groß: So gibt es sogar Befehle und Funktionen, mit denen ein Lichtgriffel oder Joystick überwacht werden kann. Doch ein Peripheriegerät, das in jüngster Zeit immer mehr an Bedeutung gewinnt, wird nicht durch eingebaute Routinen unterstützt: Die Maus.

Guide'. Mit Hilfe des Programmes MAUSFUNC.DEM (*Bild 2*) kann der Einsatz und die Art der Aufrufe mit den Parameterübergaben an die Prozeduren nachvollzogen werden. Die Prozeduren sind trotz Basic

strukturiert programmiert. Mit ihnen kann man den Mauscursor aus- und einblenden, die Mausgeschwindigkeit einstellen und sie initialisieren.

Im Folgenden werden die einzelnen Prozeduren mit Namen und Parametern aufgeführt und in ihrer Funktion kurz beschrieben:

Mausaufruf (m1%, m2%, m3%, m4%, m5%)
Der Maustreiber wird über den MS-DOS Interrupt 51 aufgerufen.

MausInit (Maus%, NrMausButton%)
Die Prozedur initialisiert und testet eine

```

SUB MausAufruf ( m1%, m2%, m3%, m4%, m5% )

    Aufruf des MS-DOS Interrupts 51

    REG %ax, m1%
    REG %bx, m2%
    REG %cx, m3%
    REG %dx, m4%
    REG %es, m5%

    CALL INTERRUPT 51

    m1% = REG (%ax)
    m2% = REG (%bx)
    m3% = REG (%cx)
    m4% = REG (%dx)

END SUB

SUB MausInit (Maus%, NrMausButton%)

    Initialisiere und teste ob Maus vorhanden ist

    SHARED MausCurFlag%

    Maus% = %False
    MausCurFlag% = %False
    CALL MausAufruf ( Maus%, NrMausButton%, 0, 0, 0 )

END SUB

SUB ShowMaus

    Zeige den Maus-Cursor

    SHARED MausCurFlag%

    IF NOT MausCurFlag% THEN
        CALL MausAufruf ( 1, 0, 0, 0, 0 )
        MausCurFlag% = %True
    END IF

END SUB

SUB HideMaus

    Blende Maus-Cursor aus

    SHARED MausCurFlag%

    IF MausCurFlag% THEN
        CALL MausAufruf ( 2, 0, 0, 0, 0 )
        MausCurFlag% = %False
    END IF

END SUB

SUB GetMausPos ( Left%, Both%, Right%, xMausPos%, yMausPos% )

    Hole die Mausposition und Tastendruck

    LOCAL Button%

    CALL MausAufruf ( 3, Button%, xMausPos%, yMausPos%, 0 )

    Both% = (Button% AND 3) = 3
    IF (NOT Both%) THEN
        Left% = (Button% AND 1) = 1
        Right% = (Button% AND 2) = 2
    END IF

END SUB

SUB SetMausPos (xMausPos%, yMausPos%)

    Positioniere die Maus

    CALL MausAufruf ( 4, 0, xMausPos%, yMausPos%, 0 )

END SUB

```

Bild 1. Mit dieser Include-Datei wird der Einsatz der Maus unter Turbo-Basic einfach

```

' Setze Minimum und Maximum für die vertikale Cursorposition
CALL MausAufruf ( 8, 0, Min%, Max%, 0 )

```

```
END SUB
```

```
SUB SetMausIcon(HotSpot%, yHotSpot%, Cursor%(1))
```

```
' Setze Form des grafischen Maus-Cursors
```

```
LOCAL m4%, m5%
```

```
m4% = VARPTR(Cursor%(0,0))
```

```
m5% = VARSEG(Cursor%(0,0))
```

```
CALL MausAufruf ( 9, xHotSpot%, yHotSpot%, m4%, m5% )
```

```
END SUB
```

```
SUB SetMausTextIcon(ScrMask%, CurMask%)
```

```
' Setze Form des Text-Maus-Cursors
```

```
CALL MausAufruf ( 10, 0, ScrMask%, CurMask%, 0 )
```

```
END SUB
```

```
SUB ReadMausMotionCounters(xMausPos%, yMausPos%)
```

```
' Lese Mickey-Zähler
```

```
CALL MausAufruf ( 11, 0, xMausPos%, yMausPos%, 0 )
```

```
END SUB
```

```
SUB SetMausIRQ(CallMaske%, MArray%(1))
```

```
' Setze Adresse der Interrupt-Routine und Ereignis
```

```
LOCAL m4%, m5%
```

```
m4% = VARPTR(MArray%(0))
```

```
m5% = VARSEG(MArray%(0))
```

```
CALL MausAufruf ( 12, 0, CallMaske%, m4%, m5% )
```

```
END SUB
```

```
SUB GetMausButtonPressed(Left%, Both%, Right%, Button%, xMausPos%, yMausPos%)
```

```
' Hole Testeninformationen beim Drücken
```

```
IF Left% THEN
```

```
Button% = 1
```

```
ELSEIF Right% THEN
```

```
Button% = 2
```

```
ELSE
```

```
Button% = 3
```

```
END IF
```

```
CALL MausAufruf ( 5, Button%, xMausPos%, yMausPos%, 0 )
```

```
Both% = (Button% AND 3) = 3
```

```
IF (NOT Both%) THEN
```

```
Left% = (Button% AND 1) = 1
```

```
Right% = (Button% AND 2) = 2
```

```
END IF
```

```
END SUB
```

```
SUB GetMausButtonReleased(Left%, Both%, Right%, Button%, xMausPos%, yMausPos%)
```

```
' Hole Testeninformationen beim Loslassen
```

```
IF Left% THEN
```

```
Button% = 1
```

```
ELSEIF Right% THEN
```

```
Button% = 2
```

```
ELSE
```

```
Button% = 3
```

```
END IF
```

```
CALL MausAufruf ( 6, Button%, xMausPos%, yMausPos%, 0 )
```

```
Both% = (Button% AND 3) = 3
```

```
IF (NOT Both%) THEN
```

```
Left% = (Button% AND 1) = 1
```

```
Right% = (Button% AND 2) = 2
```

```
END IF
```

```
END SUB
```

```
SUB SetMausRangeX (Min%, Max%)
```

```
' Setze Minimum und Maximum für die horizontale Cursorposition
```

```
CALL MausAufruf ( 7, 0, Min%, Max%, 0 )
```

```
END SUB
```

```
SUB SetMausRangeY (Min%, Max%)
```

```

Hole Puffergröße, die der Maus-Driver benötigt
CALL MausAufruf ( 21, MausBufferSize%, 0, 0, 0 )
END SUB

SUB SaveMausBuffer(MausBufferArray%(1))
    Freie Maus-Status-Puffer
    LOCAL m4%, m5%
    m4% = VARPTR(MausBufferArray%(0))
    m5% = VARPTR(MausBufferArray%(1))
    CALL MausAufruf ( 22, 0, 0, m4%, m5% )
END SUB

SUB RestoreMausBuffer(MausBufferArray%(1))
    Stelle Maus-Status-Puffer wieder her
    LOCAL m4%, m5%
    m4% = VARPTR(MausBufferArray%(0))
    m5% = VARPTR(MausBufferArray%(1))
    CALL MausAufruf ( 23, 0, 0, m4%, m5% )
END SUB

SUB SetMausScreen(MausScreen%)
    Setze aktuelle Bildschirmseitennummer
    CALL MausAufruf ( 29, MausScreen%, 0, 0, 0 )
END SUB

DEF FNGetMausScreen%
    Hole aktuelle Bildschirmseitennummer
    LOCAL x%
    CALL MausAufruf ( 30, x%, 0, 0, 0 )
    FNGetMausScreen% = x%
END DEF

```

```

SUB SetMausLighter
    Schalte Lichtgriffel-Emulations-Modus ein
    CALL MausAufruf ( 13, 0, 0, 0, 0 )
END SUB

SUB ResetMausLightPen
    Schalte Lichtgriffel-Emulations-Modus aus
    CALL MausAufruf ( 14, 0, 0, 0, 0 )
END SUB

SUB SetMausMickeyPixelRatio(xMausRatio%, yMausRatio%)
    Setze das Verhältnis Mickey / Pixel
    CALL MausAufruf ( 15, 0, xMausRatio%, yMausRatio%, 0 )
END SUB

SUB SetMausThreshold(Threshold%)
    Setze Doppelte-Geschwindigkeits-Schwelle des Mauscursors
    CALL MausAufruf ( 19, 0, 0, Threshold%, 0 )
END SUB

SUB SwapMausIRQ(CallMask%, MArray%(1))
    Vertausche Adressen der Interrupt-Routine und IRQ-Ereignisse
    LOCAL m4%, m5%
    m4% = VARPTR(MArray%(0))
    m5% = VARPTR(MArray%(1))
    CALL MausAufruf ( 20, 0, CallMask%, m4%, m5% )
END SUB

SUB GetMausBuffer(MausBufferSize%)

```


Microsoft Maus. Findet sich in der Variablen Maus% der Wert %True, so ist die Maus vorhanden, bei %False wurde keine Maus gefunden. NrMausButton% gibt die Anzahl der Maustasten minus Eins an.

Maus-Cursor aus- und einblenden

ShowMaus

Der Maus-Cursor wird angezeigt. MausCurFlag% wird auf %True gesetzt. Damit wird verhindert, daß die Maus mehrmals angezeigt wird.

HideMaus

Der Maus-Cursor wird, wenn er sichtbar war, wieder ausgeblendet. MausCurFlag% wird auf %False gesetzt.

GetMausPos (Left%, Both%, Right%, xMausPos%, yMausPos%)

Neben der aktuellen Position der Maus wird auch festgestellt, welche Taste gedrückt wurde. Das dürfte die Funktion sein, die am häufigsten benötigt wird.

SetMausPos (xMausPos%, yMausPos%)

Die Maus wird an die entsprechende Position gesetzt.

GetMausButtonPressed (Left%, Both%, Right%, Button%, MausPos%, yMausPos%)

Die Informationen der gedrückten Taste wird geholt.

GetMausButtonReleased (Left%, Both%, Right%, Button%, xMausPos%, yMausPos%)

Die Tasteninformationen beim Loslassen der Taste wird geholt.

Maus und Fenstertechnik

SetMausRangeX% (Min%, Max%)

Damit lassen sich Minimum und Maximum der horizontalen Cursorposition setzen. Somit kann ein Fenster vorgegeben werden, in dem sich die Maus bewegen kann.

SetMausRangeY% (Min%, Max%)

Dies ist die Prozedur für das Setzen des Minimum und Maximum der vertikalen Cursorposition, also für die Y-Koordinaten eines Fensters.

SetMausIcon (xHotSpot%, yHotSpot%, Cursor%(2))

Die Form des grafischen Maus-Cursors (Icon) wird als zweidimensionales Array übergeben. Die Anordnung des Arrays und die Einleseprozedur können im Programm MAUSFUNC.DEM studiert werden.

den. Diese Prozedur kann natürlich öfters mit verschiedenen Bildsymbolen aufgerufen werden.

SetMausTextIcon (ScrMask%, CurMask%)

Mit diesem Aufruf läßt sich die Form des Text-Maus-Cursors festlegen.

ReadMausMotionCounters (xMausPos%, yMausPos%)

Die Anzahl der Schritte, die die Maus seit dem letzten Auslesen zurückgelegt hat, wird übergeben (Mickey-Zähler).

SetMausIRQ (CallMaske%, MLArray%(1))

Diese Prozedur setzt die Adresse der Interrupt-Routine in Zuordnung zu einem Ereignis. Im MLArray ist die Interrupt-routine abgespeichert (→ Turbo-Basic-Handbuch, Seite 211). Mit dieser Funktion und dem Aufruf SwapMausIRQ können eigene Interruptprogramme eingebunden werden. Sie werden beim Eintritt eines bestimmten Ereignisses, das in CallMaske definiert wird, aufgerufen.

SetMausLightPen

Der Aufruf bewirkt das Einschalten des Lichtgriffel-Emulations-Modus.

ResetMausLightPen

Der Lichtgriffel-Emulations-Modus wird ausgeschaltet.

Einstellung der Mausgeschwindigkeit

SetMausMickeyPixelRatio (xMausRatio%, yMausRatio%)

Mit dieser Prozedur kann das Verhältnis Mickey/Pixel, d. h. zwischen tatsächlicher Mausbewegung auf dem Tisch und der Bewegung des Maus-Cursors auf dem Bildschirm eingestellt werden.

SetMausThreshold (Threshold%)

Die Prozedur setzt die Schwelle für den Einsatz der doppelten Geschwindigkeit des Maus-Cursors.

SwapMausIRQ (CallMaske%, MLArray%(1))

Die Adressen der Interrupt-Routine und IRQ-Ereignisse werden vertauscht (→ SetMausIRQ).

GetMausBuffer (MausBufferSize%)

Die vom Maus-Driver benötigte Puffergröße wird geholt.

SaveMausBuffer (MausBufferArray%(1))

Der Maus-Status-Puffer wird zwischengespeichert. MausBufferArray muß mit

MausBufferSize% dimensioniert und wird normalerweise als DYNAMIC deklariert sein. Mit dieser und der folgenden Prozedur läßt sich der Maus-Status retten. Das wird nötig, wenn ein anderes Programm aufgerufen wird, das ebenfalls die Maus anspricht.

RestoreMausBuffer (MausBufferArray%(1))

Der Maus-Status-Puffer wird auf den geretteten Inhalt gebracht – natürlich muß MausBufferArray vor diesem Aufruf mit SaveMausBuffer gerettet worden sein.

SetMausScrNo (MausScrNo%)

Der Aufruf setzt die aktuelle Bildschirmseitennummer.

GetMausScrNo (MausScrNo%)

Die aktuelle Bildschirmseitennummer wird geholt.

Konventionen

Im Hauptprogramm müssen einige Konstanten definiert sein, bevor man die Maus-Prozeduren aufrufen kann. Das sind zum einen die Register (Übergabe der Parameter über Register):

```
%ax = 1:
%bx = 2:
%cx = 3:
%dx = 4:
%si = 5:
%di = 6:
%es = 9
```

Zum anderen werden die boolischen Wahrheitswerte definiert (die auch für andere Anwendungen nützlich sein können):

```
%True = -1
%False = 0
```

Initialisierung der Maus

Auf eine Variable wird von mehreren Routinen zugegriffen. Sie ist deshalb in den Prozeduren als SHARED deklariert und muß im Hauptprogramm definiert sein: MausCurFlag% = %False zeigt zunächst an, daß der Maus-Cursor ausgeschaltet ist; die Routinen zur Anzeige des Maus-Cursors ändern den Inhalt des Flags in %True.

Um die Maus in einem Programm ansprechen zu können, muß die Prozedur MausInit aufgerufen werden. Die restlichen Funktionen können in einer der Programmstruktur entsprechenden Reihenfolge eingesetzt werden. □

DIE TURBO-SERIE

Auch auf Diskette



Turbo-Pascal, Ausgabe 9
Best.-Nr. 0972, 28,- DM/str., 230,- €S

Professionelle Programmierer bieten in diesem Heft professionelle Anwendungen für diese am weitesten verbreitete höhere Programmiersprache. Alle Programmierideen lassen sich auf allen MS-DOS und CP/M-Computern anwenden.

Außerdem zum gleichen Thema, aus der CHIP WISSEN Reihe:

Programmieren mit Turbo-Pascal
Von Thomas Glese

Dieses Buch führt im ersten Teil anhand vieler praktischer Beispiele in die Grundelemente von Turbo-Pascal ein. Übungsaufgaben mit Lösungen erleichtern den Einstieg. Als komplexes Anwendungsbeispiel wird die Programmierung eines Schachspiels behandelt. Darüber hinaus enthält das Buch verschiedene Grafik-Listings.

160 Seiten, 70 Abb., DM 30,-
ISBN-Nr. 3-8033-0190-0

Alle Programme aus den CHIP SPECIAL Turbo-Pascal 1-9 erhalten Sie auf Diskette für folgende Disketten-Formate: 3", 3,5", 5,25" und 8". Die Programme laufen auf allen MS-DOS und CP/M Rechnern. Die Auslieferung der Software erfolgt über den Elsa-Data CHIP Shop. Bitte beachten Sie, daß die Dokumentation zu den Disketten in der entsprechenden CHIP SPECIAL Ausgabe zu finden ist.

SOFORT BESTELLEN

BESTELLCOUPON:

Bitte ausfüllen, unterschreiben und einsenden an CHIP-Leser-Service 731, Vogel-Verlag, Postfach 6740, D-8700 Würzburg 1

Ja, bitte liefern Sie mir die angekreuzten Produkte zu den genannten Preisen plus Versandkosten

Anzahl		Best.-Nr.	Einzelpreis DM/str. €S	
	Turbo-Pascal SPECIAL 1	0120	28,-	230,-
	Turbo-Pascal SPECIAL 2	0310	28,-	230,-
	Turbo-Pascal SPECIAL 3	0400	28,-	230,-
	Turbo-Pascal SPECIAL 4	0460	28,-	230,-
	Turbo-Pascal SPECIAL 5	0560	28,-	230,-
	Turbo-Pascal SPECIAL 6	0580	28,-	230,-
	Turbo-Pascal SPECIAL 7	0610	28,-	230,-
	Turbo-Pascal SPECIAL 8	0948	28,-	230,-
	Turbo-Pascal SPECIAL 9	0972	28,-	230,-

Datum, Unterschrift

Vorname, Name

STRASSE, Nr.

PLZ, Ort

Die Lieferung der Special erfolgt gegen Rechnung plus Versandkostenanteil.



Erich Esders

Bäumchen wechsle dich

Folgendes Problem steht zur Lösung: Zu bestimmen ist diejenige 9stellige Ganzzahl, in der jede Ziffer von 1 bis 9 genau einmal vorkommt und für die gilt: Für alle k zwischen 1 und 9 müssen die ersten k Ziffern als eigenständige Zahl genommen ohne Rest durch k teilbar sein. Eine erste Überlegung ergibt, daß es genau $9! = 362880$ Zahlen auf das Teilbarkeits-Kriterium zu untersuchen gilt. Es sind dies alle Permutationen, die die 9 unterschiedlichen Ziffern einnehmen können. Es ist auch unmittelbar einleuchtend, daß für die Ziffer 5 nur die fünfte Stelle in der zu suchenden Zahl in Frage kommt, denn alle Vielfache von 5 enden immer mit 0 oder 5, und die Ziffer 0 gehört ja nicht zur oben angegebenen Menge. Aber spätestens an dieser Stelle taucht die Frage auf, ob es eventuell nicht mehrere Lösungen gibt und ob ein Rechner mit geeignetem Programm diese nicht auch finden könnte.

Entwirft man nun aber ein Programm, so ist es sicherlich nicht weiter aufwendig, auch gleich die anderen Ziffernanzahlen zwischen 1 und 9 auf das oben geschilderte Kriterium zu untersuchen. Eine mögliche Lösung zeigt das C-Programm DIVNUM.C, das in Bild 1 gezeigt ist. Eine Sprache wie C bietet sich aufgrund des weiter unten geschilderten, stark rekursiven Permutations-Algorithmus an.

Abbruch bei unzulässigen Parametern

Für das Betriebssystem kommt die Ausführung eines Benutzerprogramms dem Aufruf einer Subroutine gleich, eventuell übergebene Parameter sind die Variablen `argc` und `*argv[]`. Erstere spiegelt die Anzahl aller einzelnen Worte der Eingabezeile wieder, letztere ist ein Array bestehend aus Zeigern, die jeweils auf ein solches Wort weisen. Da mit MS-DOS ab der Version 3.00 der Programmname mitgezählt wird, enthält `argc` mindestens eine 1, bei Angabe eines weiteren Parameters dann 2. Allgemein kann man dann durch `*argv[argc-1]` für alle Werte von `argc` vom aktuellen bis hinunter zu 1 auf die einzelnen Worte zugreifen [2].

Permutationen werden in den unterschiedlichsten Zusammenhängen gebraucht. Am Beispiel einer Knobelaufgabe, die zusätzliche Bedingungen an die aus den Ziffern 1 bis 9 gebildeten Zahlen stellt, wird eine programmtechnische Lösung in der Sprache C entwickelt.

In `main` bearbeitet das Programm zunächst einmal den optionalen Parameter und bricht die Ausführung bei ungültigen Werten ab. Bei gültiger Digit-Anzahl wird das Array `number[9]` mit den fortlaufenden Digits 1 bis 9 als Ausgangszahl initialisiert und `permutate` mit

```

/*****
 * DIVNUM.C                               Erich Esders
 *
 * Bestimmt die n-stellige Ganzzahl, in der jede Ziffer von
 * 1 bis n nur einmal vorkommt und für die gilt:
 *
 * Für alle k zwischen 1 und n müssen die ersten k Ziffern als
 * eigenständige Zahl genommen ohne Rest durch k teilbar sein!!
 *
 * Default für den Programmlauf ist n = 9; dieser Standard kann
 * durch Angabe der einstelligen Aufruf-Option 'number' in Form
 * einer gültigen Ziffer { 1, 2, ..., 9 } geändert werden.
 *
 * COMPILE : MSC divnum.c /Ot;             MICROSOFT C-Compiler V4.00
 * LINK     : LINK divnum;                 /Ot : Code zeitoptimiert
 * RUN      : divnum {number}              MICROSOFT Linker V3.51
 *                                              MS-DOS V3.20
 *****/

#include <stdio.h> /* Standard I/O-Prozeduren */

long atol( int ); /* Array-To-Long Conversion */
int main( int, char *[] ); /* Hauptprogramm */
void permutate( int ); /* array element permutation */
void test_div( void ); /* Divisions-Test auf Rest */

int n_dig = 9; /* Anzahl gewünschter Digits */
int n_res = 0; /* Ergebnis-Zähler */
char number[9]; /* Digit-Array der Ganzzahl */

/* -----
   name : main( argc, argv )
   Job  : Ganzzahl gemäß Spezifikation bzw. Option
         initialisieren, untersuchen und eventuelle
         Ergebnisse auf CON: ausgeben;
   Input : optionales argv[1] = Anzahl Digits der Ganzzahl;
   Output : 0 = Lauf OK, 1 = Fehlerhafte Option;
   Calls : permutate( k ), printf( ctrl, var );
   ----- */

int main( argc, argv )
int argc;
char *argv[];
{
    int i;

    if ( argc == 2 ) /* optionalen Parameter holen */
        n_dig = *argv[ argc-1 ] - '0';

    if ( n_dig < 1 || n_dig > 9 )
        { printf( "Anzahl Digits unzulässig!! { 1,2,...,9 }\r\n\007" );
          exit( 1 );
        }

    else
        { for ( i = 1; i <= n_dig; i++ ) number[i] = i;
          permutate( n_dig );
          if ( !n_res )
              printf( "Keine Lösung(en) für die Ziffern 1 bis %i !!!\r\n", n_dig );
          exit( 0 );
        }
}

```

Bild 1. Das C-Programm DIVNUM.C


```

/* -----
Name   : permutate( k )
Job    : Array-Elemente "number[]" permutieren und die
        nächste Permutation auf Divisionsrest testen;
Input  : k = INTEGER-Index des nächsten Array-Elements,
        das zur Permutation ansteht;
Output : /
Calls  : test_div(), permutate( K );
----- */

void permutate( k )
int k;

{ int i, x;
  if ( k == 1 )
    test_div();
  else
    { permutate( k-1 );
      for ( i = 1; i < k; i++ )
        { x = number[i]; number[i] = number[k]; number[k] = x;
          permutate( k-1 );
          x = number[i]; number[i] = number[k]; number[k] = x;
        }
    }
}

/* -----
Name   : test_div()
Job    : aktuellen Inhalt des Arrays "number[]" als Ganz-
        zahl interpretieren und die ersten n Digits
        auf Teilbarkeit durch n untersuchen;
Input  : /
Output : /
Calls  : atol( 1 ), cprintf( ctrl, var );
----- */

void test_div()
{ int i = 1,
    rem = 0;

  while( i++ < n_dig && !rem )
    rem = atol( i ) % i;

  if ( i > n_dig && !rem )
    { cprintf( "%i. mögliche Lösung : ", ++n_res );
      for ( i = 1; i <= n_dig; i++ ) cprintf( "%i", number[i] );
      cprintf( "\r\n" );
    }
}

/* -----
Name   : long atol( k )
Job    : Wandelt die ersten k Elemente des Arrays "number[]"
        in eine LONG INTEGER;
Input  : k = Anzahl der ersten Elemente für die Wandlung;
Output : Resultat als LONG INTEGER;
Calls  : /
----- */

long atol( k )
int k;

{ long result = 0;
  int i = 1;

  while( k-- )
    result = 10*result + number[i++];

  return( result );
}

```

```

C:\>divnum 0
Anzahl Digits unzulässig!!! { 1,2,...,9 }

C:\>divnum 3
1. mögliche Lösung : 123
2. mögliche Lösung : 321

C:\>divnum 8
Keine Lösung(en) für die Ziffern 1 bis 8 !!!

C:\>divnum
1. mögliche Lösung : 381654729

```

Bild 2. Die Problemlösung ist von k zu k recht unterschiedlich

der Bestimmung aller möglichen Permutationen beauftragt. Sind keine Lösungen für die gewünschte Digit-Anzahl bestimmt worden, so beendet DIVNUM seinen Lauf mit entsprechender Meldung.

Die Funktion permutate entspricht einem Algorithmus aus [1] und erzeugt nach ihrem Aufruf alle möglichen Permutationen von Array-Elementen der übergebenen Anzahl. Eine nächste Permutation ist genau dann bestimmt, wenn diese Anzahl 1 ist, und wird dann test_div zur Überprüfung aller Teilzahlen auf Division ohne Rest durchgeführt. Die Funktion ist stark rekursiv, da sie sich selbst zweimal aufruft, um die nächst niedrigere Permutation zu bestimmen.

Eine Liste aller Permutationen

Die Reaktionen des Programms auf einige unterschiedliche Aufrufe zeigt Bild 2. Diesem kann nun auch die Lösung des eingangs geschilderten Problems entnommen werden. Der Leser möge die entstandene Zahl entsprechend überprüfen. Eine abschließende Bemerkung zur Laufzeit: Sie beträgt auf einem 10-MHz-AT für einen Aufruf divnum bzw. divnum 9 etwa 80 s, wobei das endgültige Ergebnis als 1. Lösung bereits nach 3 s bestimmt ist.

Ersetzt man in permutate den Aufruf test_div(); durch die Zeile

```
for ( i = 1; i <= n_dig; i++ ) cprintf( "%i", number[i] );
```

und löscht in main die Anweisung

```
if ( !n_res ) cprintf( ... );
```

so erzeugt ein Programmaufruf divnum n eine Liste aller n-Permutationen.

Literatur

- [1] Wirth, Niklaus: Algorithmen und Datenstrukturen.
- [2] Microsoft C-Compiler, User's Manual.



Michael Collet

EGA auf 24-Nadel-Drucker

Eine Hardcopy-Routine in Turbo-Pascal

Der NEC P6 wird im Grafikbetrieb wahlweise mit 180 oder 360 Punkten pro Zoll angesteuert. Die hohe Auflösung eignet sich vornehmlich zur gleichmäßigen Schwärzung beim Bedrucken von Overhead-Folien. Der Ausdruck erfolgt um 90 Grad gedreht, damit das zur Verfügung stehende DIN-A4-Format optimal genutzt wird. Die Routine wurde in Turbo-Pascal gemäß den Strukturierungs-ideen dieser Sprache geschrieben. Sie kann, wenn sie als hcopy.inc abgespeichert wird, mit der Compiler-

Für verzerrungsfreie Bildschirmausdrucke der hochauflösenden Grafikmodi einer IBM-PC-EGA-Karte (640 × 350 Bildpunkte) sorgt ein kleines Programm. Als Drucker ist der NEC P6 vorgesehen; mit geringen Anpassungen ist die Routine auch auf anderen 24-Nadel-Druckern einsetzbar.

senen Werte für dpi sind 180 und 360, bei anderen Werten wird die Prozedur ohne Ausdruck beendet.

Die Prozedur hardcopy besteht im wesentlichen aus den zu ihr lokalen Prozeduren make_line, eight_to_24 und print_line, die in einer Schleife 80 mal abgearbeitet werden. Die Schachte-

lung der Prozeduren und Funktionen ist in Bild 1 dargestellt. Die Prozedur make_line liest jeweils eine 8 Bit breite Spalte des Video-RAM aus. Make_line ruft dazu die Funktion make_byte auf. Die Funktion ma-

lung der Prozeduren und Funktionen ist in Bild 1 dargestellt. Die Prozedur make_line liest jeweils eine 8 Bit breite Spalte des Video-RAM aus. Make_line ruft dazu die Funktion make_byte auf. Die Funktion ma-

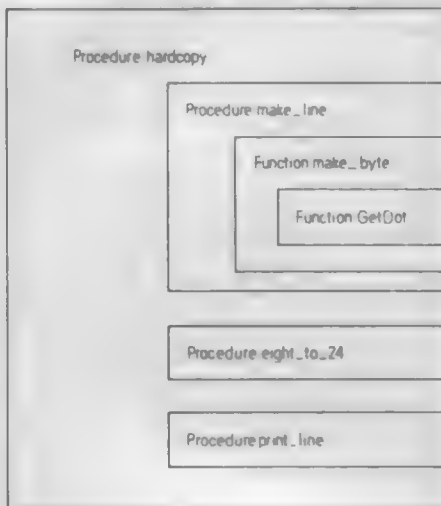


Bild 1. Die Schachtelung der Prozeduren und Funktionen

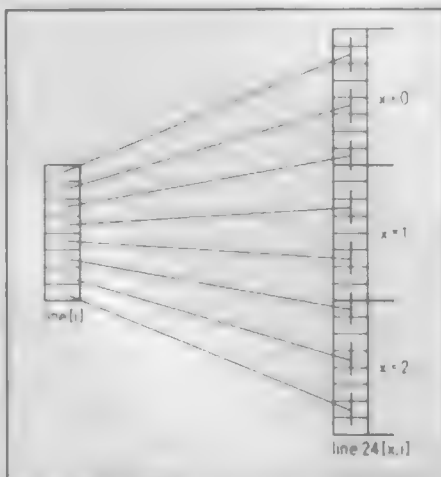


Bild 2. Die Prozedur eight_to_24 erzeugt aus einem Byte 24 Druckerpunkte

```
{ Hardcopy-Listing zum Artikel "EGA-Copy mit NEC P6 in Turbo-Pascal" }
{ Autor : Michael Collet }
{ Eickendorfer Str. 35 }
{ 2800 Bremen 1 }

PROCEDURE hardcopy(dpi:INTEGER); { dpi = Druckerauflösung }
VAR line : ARRAY [0..349] OF BYTE; { wahlweise mit 180/360 }
    count0: INTEGER; { Punkten pro Zoll }
    line24: ARRAY [0..2,0..349] OF BYTE;

PROCEDURE make_line(column:INTEGER); { liest 8 Bit=350 Zeilen }
VAR count : INTEGER; { als 1 Spalte des Video }
    { RAM }

FUNCTION make_byte(column,row:INTEGER):BYTE;
VAR count0: INTEGER;
    pixel : INTEGER;
    print : BYTE;

CONST
    bits : ARRAY [0..7] OF BYTE=(128,64,32,16,8,4,2,1);

FUNCTION GetDot (x,y : INTEGER):INTEGER;
TYPE RegPack = RECORD
    ax,bx,cx,dx,bp,si,di,ds,es,flags : INTEGER;
END;
VAR Register : RegPack;
CONST VideoInt = $10;
BEGIN
    WITH Register DO
        ax := $0d00; {ah: Pixel-Lesen}
        bx := 0; {Seiten-Nr}
        cx := x; {Spalten-Koordinate, 0..639}
        dx := y; {Zeilen-Koordinate, 0..349}
        INTR (VideoInt,Register);
        GetDot := LO(ax)
    END
END;

BEGIN
    print:=0;
    FOR count0:= 0 TO 7
    BEGIN
        pixel:=Getdot(column*8+count0,row);
        IF (pixel <> 0) THEN print:=(print OR bits[count0]);
    END;
    make_byte:= print;
END;

BEGIN
    FOR count:= 349 DOWNT0 0 DO
        line[349-count]:= make_byte(column,count)
    END;

PROCEDURE eight_to_24; { bildet die aus dem Video-RAM }
VAR i : INTEGER; { gelesene 8-Bit breite Punkt- }
CONST seven : BYTE = 128; { spalte in die 3*8 Bit der }
    six : BYTE = 64; { globalen Variablen "line24" }
```

Bild 3. Die Hardcopy-Routine in Turbo-Pascal

```

five : BYTE = 32; { ab und erstellt damit eine }
four : BYTE = 16; { vollständige Druckzeile. }
three : BYTE = 8;
two : BYTE = 4;
one : BYTE = 2;
zero : BYTE = 1;

BEGIN
FOR i:= 0 TO 349 DO
BEGIN
line24[0,i]:=0;
line24[1,i]:=0;
line24[2,i]:=0;
IF ((line[i] and seven) > 0) THEN line24[0,i]:=224;
IF ((line[i] and six) > 0) THEN line24[0,i]:=(line24[0,i] OR 28);
IF ((line[i] and five) > 0) THEN
BEGIN
line24[0,i]:=(line24[0,i] OR 3);
line24[1,i]:=128;
END;
IF ((line[i] and four) > 0) THEN line24[1,i]:=(line24[1,i] OR 112);
IF ((line[i] and three) > 0) THEN line24[1,i]:=(line24[1,i] OR 14);
IF ((line[i] and two) > 0) THEN
BEGIN
line24[1,i]:=(line24[1,i] OR 1);
line24[2,i]:=192;
END;
IF ((line[i] and one) > 0) THEN line24[2,i]:=(line24[2,i] OR 60);
IF ((line[i] and zero) > 0) THEN line24[2,i]:=(line24[2,i] OR 7);
END;
END;

PROCEDURE print_line;
VAR count1, count2, count3 : INTEGER;
iteration : INTEGER;
BEGIN
IF dpi=360 THEN
BEGIN Write(Lst, #27, '#', #40, #240, #10); { Auswahl Grafikmodus }
iteration:=7; { mit 360 dpi }
END;
ELSE
IF dpi=180 THEN
BEGIN Write(Lst, #27, '#', #39, #120, #5); { Auswahl Grafikmodus }
iteration:=3; { mit 180 dpi, schneller }
END;
FOR count1:= 0 TO 349 DO
FOR count3:=0 TO iteration DO
FOR count2:= 0 TO 2 DO
Write(Lst, char(line24[count2,count1]));
WriteLn(Lst);
END;
END;
BEGIN
IF (dpi=180 OR dpi=360) THEN
BEGIN
Write(Lst, #28, '#', #10, #10, #28, #81, #48); { Zeilenabstand 48/360 }
FOR count:= 0 TO 79 DO { fuer NEC P6, P7 etc. }
BEGIN
make_line(count);
eight_to_24;
print_line;
END;
END;
END;
END;

```

ke_byte ruft wiederum die lokal deklarierte Funktion GetDot auf, die über den jeweiligen Farbwert ermittelt, ob ein Bildpunkt gesetzt ist. Diese Funktionen liefern nur im Grafikmodus (\$10) sinnvolle Ergebnisse; in der Textbetriebsart werden zur Farbdarstellung nicht benutzte Grafikseiten als Zeichensatz-Speicher eingesetzt und auf dem Drucker erscheint nichts weiter als undefinierter „Fliegendreck“.

Ausführliche Hinweise auf Funktionsweise und Aufbau der EGA-Grafik finden sich in einer Artikelserie in mc 1/87 bis 4/87.

Die ermittelte Bildschirmspalte wird in der Variablen line gespeichert. Sie muß jetzt in ein dem 24-Nadel-Modus des Druckers bekömmlisches Format umgewandelt werden. Der NEC P6 erwartet im Grafikbetrieb jeweils 3 Byte für eine Punktspalte, jedes Bit steuert eine Drucknadel. Diese Umsetzung besorgt die Prozedur eight_to_24. Bild 2 zeigt die Zuordnung eines Byte aus der Variablen line zu den entsprechenden 3 Byte der Variablen line24.

Die Prozedur print_line initialisiert den NEC P6 auf Grafikbetrieb und übergibt die im Array line24 berechneten Werte an den Drucker. Je nach eingestellter Auflösung werden die vorliegenden Punktspalten 4- oder 8-fach ausgedruckt. Ein Punkt des EGA-Bildschirms wird somit durch eine 3x4- bzw. 3x8-Matrix auf das Papier abgebildet. Das gewährleistet gute Sicht- und Lesbarkeit von Meßwerten, Beschriftungen und Skalierungen.

Literatur

Cebulla, Ulrich: Grafik mit der IBM-EGA, mc 1/87 bis 4/87.

Die als CAD/CAM-System-Entwickler und -Hersteller bekannte HNA-Dataport-Gruppe trat mit der Vorstellung ihres neuesten Produktes CAD400 anlässlich der Aktion „IBM-Mittelstands-Expreß“ zum ersten Mal als autorisierter Vertriebspartner (Gebiet) von IBM auf.

Gegründet wurde die HNA-Dataport-Gruppe 1981, bekannt wurde sie mit dem CAD/CAM-Systemen CAD190 und CAD200. Das Unternehmen kann im deutschsprachigen Raum mehr als 700 und weltweit über 3000 Installationen vorweisen.

CAD400 läuft auf dem RISC-System 6150 von IBM. Dataport hält dieses System, nach der Aussage des Geschäftsführers der Zentrale Dataport Wolfgang Schenk, für zukunftsweisend für den mittelständigen Bereich. In der Entscheidung, eine eigene Vertriebsgruppe aufzubauen, die ausschließlich IBM-Systeme für das Programm

CAD400 von Dataport auf IBM 6150

CAD400 anbietet, sei man bestärkt worden durch erfolgreiche Ansätze der gemeinsamen Vertriebsbemühungen und die Fähigkeit von IBM, als Hardware-Lieferant einschließlich der erforderlichen Peripherie alles aus einem Guß zu bieten. Eine zweite Vertriebseinheit operiert parallel dazu für die HP-Welt (HP9000 Serie 300).

Das System 6150 von IBM zeichnet sich besonders aus durch:

– einen RISC-Prozessor neuester Technologie mit 100 ns Zykluszeit

- integrierten Gleitkommaprozessor Motorola MC 68881 (20 MHz getaktet)
- schnellen Hauptspeicher mit 100 ns Zugriffsgeschwindigkeit
- Festplatten mit höheren Transferraten
- direkt anschließbare Grafikbildschirme bis 19 Zoll mit hoher Auflösung (1024 x 1024)
- Programmiersprachen: C, Pascal, Fortran und weitere.

CAD400 operiert auf Unix-Basis und wurde in der Programmiersprache C geschrieben, es ist eine modular einsetzbare integrierte Programmfamilie.

Es wendet sich speziell an Fertigungsbetriebe und Planungsgruppen für die Anwendungslösungen: mechanische Konstruktion mit NC-Programmieranforderungen, E-Technik, Anlagenbau, Bauplanung, Design, Möbel- und Ladenbau sowie Einrichtungsplanung.

br

Karl-Heinz Schulz

Handbremse für den AT

PC-Programme langsamer gemacht

Für unter MS-DOS ladbare Programme habe ich hier eine Lösung gefunden: das Programm PCSPEED (Bild). Hiermit läßt sich in etwa die PC-Geschwindigkeit auf einem AT simulieren. Dazu benutzt das Programm den Timer-Interrupt (INT 8 H) an den es sich resident anhängt. Dieser Interrupt erfolgt automatisch ca. alle 55 msec und führt jetzt zusätzlich das Programm PCSPEED aus.

Es wird beim Erstaufruf (Installation) mit einem Wert zwischen 1 und 65535 aufgerufen. Bei jeder anderen Eingabe (z. B.: fehlender Parameter, Buchstaben, zu großer Wert) wird mit einer Fehlermeldung terminiert. Für einen AT mit 6 MHz lautet der

Während einige PC-Programme (z. B. MS-WORD) bzw. -Spiele (z. B. Flugsimulator) auf einem Olivetti M24 oder einem AT erst richtig zur Geltung kommen, gibt es einige Programme (und fast alle Grafikspiele), für die die Ausführung auf einem M24 bzw. einem AT zu schnell erfolgt.

Aufruf 'PCSPED 18000', bei 8 MHz 'PCSPEED 25000' und bei 10 MHz 'PCSPEED 34000'. Diese Werte ergeben dann in etwa eine Programmausführungsgeschwindigkeit, die dem PC entspricht.

In der Praxis hat sich jedoch gezeigt, daß die Werte etwas geringer sein sollten, da die Ausführungszeiten für Befehle beim PC

und beim AT unterschiedlich sind und durch PCSPEED auch die Reaktion auf eine Tastatureingabe verzögert werden kann. Da PCSPEED die Werte bei mehrmaligen Aufrufen bis zum Maximum (65535) addiert, kann man sich so langsam an die gewünschte Geschwindigkeit herantasten. PCSPEED

bleibt resident, aber man muß nicht neu booten, um die Verzögerung wieder herauszuwerfen. Beim zweiten oder nachfolgenden Aufrufen ist ein Aufruf mit dem Wert '0' zulässig. Dies entspricht keiner Verzögerung; PCSPEED führt keine Addition durch, sondern löscht den bisherigen (akkumulierten) Wert im Speicher und fängt tatsächlich bei Null wieder an. □

```
NAME      PCSPEED
;
; PARM_PTR EQU 30H
;
; CSEG      segment 'CODE'
;          ASSUME cs:CSEG,ds:CSEG,es:CSEG
;          ORG 100H
;          JMP INSTALL
;
; TEST      DW 3231H,3433H
;
; INT 8:    DW 0
; OLDSEG    DW 0
; ZAEHLER   DW 0
;
; LAG:      PUSHF
;          CALL DWORD PTR CS:[OLDSEG]
;          PUSH CX
;          MOV CX,CS:[ZAEHLER]
; HERE:     LOOP HERE
;          POP CX
;          IRET
;
; INSTALL:  XOR AX,AX
;          MOV SI,PARM_PTR
;          MOV CX,10
;          MOV BH,0
; REP1:     MOV BL,[SI]
;          INC SI
;          CMP BL,0
;          JZ WEITER
;          CMP BL,0
;          JB REP_ERR
;          CMP BL,9
;          JA REP_ERR
;          AND BL,0FH
;          MUL CX
;          OR DX,DX
;          JNZ REP_ERR
;          ADD AX,BX
;          JNC REP1
; REP_ERR:  JMP ERROR
;
; WEITER:   MOV ZAEHLER,AX
;          MOV AL,8
;          MOV AH,35H
;          INT 21H
;          MOV [OLDSEG],BX
;          MOV [OLDSEG],ES
;          MOV DX,ES:[BX-8]
;          CMP DX,3433H
;          JNZ CONT1
;          MOV DX,ES:[BX-10]
;          CMP DX,3231H
;          JZ CONTF
```

Ein Assemblerprogramm paßt die AT-Geschwindigkeit an.

```
CONT1:     CMP ZAEHLER,0
;          JZ ERROR
;          MOV DX,OFFSET LAG
;          MOV AL,8
;          MOV AH,25H
;          INT 21H
;          MOV DX,OFFSET INSTALL
;          MOV AL,0
;          MOV AH,31H
;          INT 21H
;          ; Segment in DS
;          ; DS:DX
;          ; Setze neue Adresse INT8
;          ; Resident bis hier
;          ; RETURN-Kode
;          ; Fn31 von INT21 == INT27
;
; Hier weiter, wenn PCSPEED
; bereits installiert ist;
; d.h. letzte Installation INT8
; Sonst wird PCSPEED nicht er-
; kannt und oben neu installiert
;
; CONTF:    CMP ZAEHLER,0
;          JNZ CONTADD
;          MOV DX,OFFSET MSG_NUL
;          CALL MSG
;          MOV DX,1
;          JMP CONTSET
;
; CONTADD:  MOV DX,OFFSET MSG_ADD
;          CALL MSG
;          MOV DX,ES:[BX-2]
;          ADD DX,ZAEHLER
;          JNC CONTSET
;          MOV DX,0
;          ; Addiere bis 0ffffH
;          ; CRY = überlauf
;          ; Maximum
;
; CONTSET:  MOV ES:[BX-2],DX
;          JMP AUSGANG
;
; ERROR:    MOV DX,OFFSET ERR_MSG
;          CALL MSG
;
; AUSGANG:  INT 20H
;
; MSG:      PUSH AX
;          PUSH BX
;          PUSH CX
;          PUSH ES
;          MOV AH,9
;          INT 21H
;          POP ES
;          POP CX
;          POP BX
;          POP AX
;          RET
;
; MSG_NUL:  DB 'PCSPEED-Schleife ruckgesetzt auf 0!',13,10,'$'
; ERR_MSG:  DB 'Parameterangabe falsch!',13,10,'$'
; MSG_ADD:  DB 'PCSPEED 1" bis "PCSPEED 65535"',13,10,'$'
;           DB 'PCSPEED ist bereits installiert!',13,10,'$'
;           DB 'Addition wird durchgeführt!',13,10,'$'
;
; CSEG      ENDS
;          END
;          PROG
```


Christoph Pahl

Aufs Ganze gehen

Schnelles Rechnen mit normalisierten Zahlen

Der Grund für die Komplexität und Langsamkeit von Fließkommaroutinen ist letztlich die Forderung, daß aus einem großen Zahlenbereich alle Werte mit derselben relativen Genauigkeit dargestellt werden sollen. Verzichtet man auf diesen Anspruch, so können reelle Zahlen im Computer auf sehr viel effizientere Art repräsentiert werden: Eine reelle Zahl r wird durch den Integeranteil i des Produkts aus r und einem konstanten Faktor F dargestellt. Die reelle Zahl errechnet sich dann (mit einem gewissen Rundungsfehler) gemäß

$$r = i / F$$

Diese Darstellung kann man sich leicht veranschaulichen: Eine Integervariable mit dem Wert x stellt nicht wie üblich die Zahl x dar, sondern x „F-tel“. Der sog. Normalisierungsfaktor F bestimmt den möglichen Zahlenbereich:

Größe darstellbare Zahl = maxint / F
Kleinste darstellbare Zahl = minint / F

Dabei bedeuten maxint den größten (positiven) Integerwert und minint den kleinsten (negativen) Integerwert. maxint und minint hängen vom Computer und der Programmiersprache ab, oft stehen auch mehrere Integer-Typen zur Verfügung.

Von F abhängig ist auch die Genauigkeit: Zwei Zahlen werden sicher unterschieden, wenn ihre Differenz den Betrag $1/F$ hat. Ist der darzustellende Zahlenbereich größer als der Integer-Wertebereich, also F kleiner 1, so ist natürlich die Genauigkeit schlechter als 1. Bei der Berechnung des Normalisierungsfaktors F für eine bestimmte Anwendung muß man sich zunächst über die auftretenden Zahlenwerte klarwerden. Um Integerüberläufe auszuschließen, muß gelten:

$F \leq \text{maxint} / \text{größte auftretende positive Zahl}$

und

$F \leq \text{minint} / \text{kleinste auftretende negative Zahl}$

Bei der Wahl des Integer-Typs, der ja maxint und minint festlegt, spielen Überlegungen bezüglich der gewünschten Genauig-

Fließkommazahlen sind für Rechner ohne Arithmetikcoprozessoren aufwendig zu verarbeiten. Für wirklich schnelle Programme empfiehlt es sich deshalb, reelle Zahlen als Integerwerte darzustellen. Die berühmt-berüchtigte Mandelbrotmenge läßt sich damit bis zu 100mal schneller berechnen.

keit eine Rolle. Treten in der Anwendung nur positive reelle Zahlen auf, so wird man vorzeichenlose Integerzahlen wählen, da

wegen ihres größeren maxint -Wertes eine höhere Genauigkeit erreicht werden kann.

Rechenregeln

Die Vorschriften zum Rechnen mit normalisierten Zahlen lassen sich leicht aus ihrer Definition ableiten (Bild 1). Der Normalisierungsfaktor F sollte eine Potenz von Zwei sein, denn dann ist Division durch F durch die schnellere arithmetische Bitver-

1. Addition und Subtraktion

Man möchte zwei reelle Zahlen r_1, r_2 addieren, das Ergebnis heiße r_3 :

$$r_3 = r_1 + r_2$$

Die r_i sind als Integerwerte i_i dargestellt mit $i_i = [r_i \cdot F]$. (Die gaußsche Klammer $[]$ rundet auf die nächste ganze Zahl ab.)

Im Rahmen der Rechengenauigkeit ist obige Addition deshalb gleichbedeutend mit

$$\frac{i_3}{F} = \frac{i_1}{F} + \frac{i_2}{F}$$

also gilt: $i_3 = i_1 + i_2$

Analog: Aus $r_3 = r_1 - r_2$ ergibt sich $i_3 = i_1 - i_2$. Man sieht: Normalisierte Zahlen lassen sich auf übliche Weise addieren und subtrahieren.

2. Multiplikation

$$r_3 = r_1 \cdot r_2 \text{ bedeutet: } \frac{i_3}{F} = \frac{i_1}{F} \cdot \frac{i_2}{F}$$

$$\text{also: } i_3 = \frac{i_1 \cdot i_2}{F}$$

Normalisierte Zahlen multipliziert man wie ganze Zahlen und teilt dann durch den Normalisierungsfaktor.

3. Division

$$r_3 = r_1 : r_2 \text{ bedeutet: } \frac{i_3}{F} = \frac{i_1}{F} : \frac{i_2}{F}$$

d. h.: $i_3 = i_1 : i_2 \cdot F$

Normalisierte Zahlen dividiert man wie ganze Zahlen und multipliziert dann mit dem Normalisierungsfaktor. (Läßt sich die Multiplikation vor der Division ausführen, so erreicht man höhere Genauigkeit.)

4. Multiplikation oder Division mit einer nichtnormalisierten ganzen Zahl n :

$$r_2 = r_1 \cdot n \text{ bedeutet: } \frac{i_2}{F} = \frac{i_1}{F} \cdot n$$

also: $i_2 = i_1 \cdot n$

Analog: Aus $r_2 = r_1 : n$ folgt $i_2 = i_1 : n$

Multiplikation und Division einer normalisierten Zahl mit einer nichtnormalisierten, ganzen Zahl erfolgt durch einfache Multiplikation bzw. Division. Auf vorherige Normalisierung kann man hier (im Gegensatz zur Addition!) also verzichten.

Bei allen Regeln ist zu beachten:

Ist $F < 1$, verwendet man statt Multiplikation mit F Division durch $\frac{1}{F}$, statt Division durch F Multiplikation mit $\frac{1}{F}$, um allein mit Integeroperationen auszukommen.

Bild 1: Rechenregeln für normalisierte Zahlen

schiebung nach rechts um den Zweierlogarithmus von F ersetzbar, Multiplikation mit F durch entsprechende Verschiebung nach links. Das Rechnen mit normalisierten Zahlen geht sehr rasch, da Mikroprozessoren Integerzahlen effizient verarbeiten (beim M68000 z. B. ist selbst für Multiplikation und Division nur ein Maschinensprachebefehl nötig).

Die begrenzte Genauigkeit ist im allgemeinen nicht störender als bei Fließkommazahlen. Der Preis für die erreichbare Geschwindigkeit besteht neben der etwas unbequemen Handhabung vor allem in der Notwendigkeit, die maximal auftretenden Werte genau bestimmen zu müssen, um Integerüberläufe zu vermeiden. Fließkommazahlen dürfen eben in einem vergleichsweise riesengroßen Bereich liegen. Das Finden der maximalen Werte stellt einen gewissen Aufwand dar, es ist meist aber doch erheblich leichter als im folgenden Anwendungsbeispiel.

Anwendung Mandelbrot-Menge

Die Mandelbrot-Menge ist berüchtigt vor allem durch ihre lange Rechenzeit. Sie ist wohl das spektakulärste Beispiel für Rechenzeitverkürzung mittels normalisierter Zahlen. Das Programm ist auf dem Atari-ST in Megamax C mit etwas Inline-Assembler geschrieben worden. Es werden einige Tips zum Umschreiben auf andere Systeme gegeben.

Zur Berechnung der Mandelbrot-Menge wird auf den Bildschirm ein Ausschnitt aus der Ebene der komplexen Zahlen projiziert, also jedem Pixel eine bestimmte komplexe Zahl k_0 zugeordnet. Eine komplexe Zahl kann man sich als Punkt einer Ebene vorstellen, seine Koordinaten heißen Realteil und Imaginärteil.

k_0 dient als Ausgangswert für eine Iteration nach folgender Vorschrift:

$$k_{n+1} = k_n^2 + k_0$$

Alle komplexen Zahlen k , die dabei endlich bleiben, gehören der Mandelbrotmenge an. Man kann zeigen, daß k dann über alle Grenzen wächst, wenn die Größe von k irgendwann 2 überschreitet. Im Programm wird eine Zahl als Element der Mandelbrotmenge betrachtet, wenn ihr Betrag nach einer bestimmten, endlichen Anzahl von Iterationen den Wert 2 noch nicht überschritten hat (man kann ja nicht bis in alle Ewigkeit iterieren!). Überschreitet eine Zahl den Wert 2 schon vorher, dann wird abgebrochen. In Abhängigkeit der Anzahl durchgeführter Iterationsschritte wird dem Bildschirm eine Farbe zugeordnet. Die Punkte, deren Zahlen nicht zur Man-

```
k_realn+1 = k_realn2 - k_imagn2 + k_real0
k_imagn+1 = 2 · k_realn · k_imagn + k_imag0
Abbruch, falls k_real2 + k_imag2 > 4
(denn der Betrag einer komplexen Zahl ist die
Wurzel aus der Summe der Quadrate von Real-
und Imaginärteil).
```

Bild 2: Iterationsvorschrift für Real- und Imaginärteil

```
real0 aus x-Koordinate des Pixels errechnen;
imag0 aus y-Koordinate des Pixels errechnen;
real = real0;
imag = imag0;
Schleife für itschritt = 1 bis ITMAX:
    realq = real * real >> FLOG;
    imagq = imag * imag >> FLOG;
    falls realq + imagq > 4 * F:
        Abbruch der Schleife;
    imag
        = (imag * real >> FLOG-1) + imag0;
    real = realq - imagq + real0;
    /* Marke P */
Ende der Schleife;
Bezeichner:
real0, imag0: Die normalisierten Werte von
                k_real0 und k_imag0
real, imag: Die im Iterationsprozeß erhaltenen
                neuen (normalisierten)
                Werte von k_real und k_imag
realq, imagq: die (normalisierten) Quadrate
                von k_real und k_imag
itschritt: Zähler der Iterationsschritte
FLOG: 2er-Logarithmus
                des Normalisierungsfaktors
ITMAX: maximale Iterationsanzahl
```

Bild 3: Ablauf der Iterationsschleife

In der Iterationsschleife gilt nach dem Abbruchkriterium stets:

$$k_imag^2 + k_real^2 \leq 4 \quad (1)$$

da nicht abgebrochen wurde. Deswegen gilt hier auch:

$$|k_real^2 - k_imag^2| \leq 4 \quad (2)$$

Denn der Betrag dieser Differenz wird maximal, wenn eines der Quadrate maximal wird, also zu 4, und das andere minimal, also zu 0. Ebenfalls aus (1) zu folgern:

$$|k_imag \cdot k_real| \leq 2 \quad (3)$$

Da k_imag und k_real der Beziehung (1) gehorchen müssen, wird der Betrag ihres Produktes nämlich maximal, wenn $|k_real| = |k_imag| = \sqrt{2}$ (auf den korrekten Beweis mit analytischen Hilfsmitteln sei verzichtet).

Der maximale Betrag der Variablen $real$ bei /* Marke P */ ist aus (2) bestimmbar (Bem.: Unter dem Betrag einer Variablen wird hier der Betrag verstanden, den sie als *Integervariable* hat; nicht der reelle Betrag, der ihr als *normalisierte Zahl* zukommt!):

$$|real| \leq 4 \cdot F + \text{maximaler Realbetrag der dargestellten Ebene} \cdot F \quad (4)$$

Der maximale Betrag von $imag$ bei /* Marke P */ ist aus (3) ersichtlich:

$$|imag| \leq 2 \cdot 2 \cdot F + \text{maximaler Imaginärbetrag der dargestellten Ebene} \cdot F \quad (5)$$

Beim ersten Eintritt in die Schleife ist der Betrag von $real$ höchstens

$F \cdot \text{maximaler Realbetrag der dargestellten Ebene}$,

der Betrag von $imag$ höchstens

$F \cdot \text{maximaler Imaginärbetrag der dargestellten Ebene}$.

Die Beziehungen (4) und (5) gelten also innerhalb der Schleife immer!

Die neben $real$ und $imag$ in der Schleife auftretenden Werte (das sind $realq$, $imagq$, $real \cdot real$, $imag \cdot imag$, $realq + imagq$, $real \cdot imag$) liegen alle in einem deutlich größeren Bereich. Diese Werte treten aber auch nicht als Operanden einer Multiplikation auf!

Bild 4: Bestimmung der maximal auftretenden Werte

delbrotmenge gehören, machen erst die eigentliche Schönheit der Grafik aus.

Im Computerprogramm muß man natürlich Realteil k_real und Imaginärteil k_imag einer komplexen Zahl k getrennt behandeln, die Iterationsvorschrift ergibt sich dann aus den Regeln zum Rechnen mit komplexen Zahlen (Bild 2). Für die Berechnung mit normalisierten Zahlen resultiert daraus der in Bild 3 gezeigte Ablauf. Ein Hinweis zur Programmiersprache C:

>> ist der Operator für arithmetische Bitverschiebung nach rechts.

>> bindet schwächer als * oder +.

$imag * real >> FLOG-1$ ist eine schnellere Berechnung für $2 * (imag * real >> FLOG)$.

Zur Bestimmung des Normalisierungsfaktors F muß man sich nun einen Überblick über die maximal auftretenden Werte verschaffen (Bild 4).

Implementierung

Weil der im Atari ST verwendete Mikroprozessor M68000 16-Bit-Worte mit einem einzigen Befehl multipliziert, liegt der Wunsch nahe, für $real$ und $imag$ den 16-Bit-Integertyp zu verwenden (beim MEGAMAX-Compiler: `int`), für die auftretenden größeren Werte (also z.B. auch für das Zwischenergebnis $real * imag$), die nicht als Operanden einer Multiplikation auftreten, den 32-Bit-Integertyp (beim MEGAMAX: `long int`). Leider ist dieses Vorgehen in reinem C nicht zu verwirklichen: Die Mul-

Multiplikation zweier int-Werte, die in M68000-Maschinensprache einen 32-Bit-Wert ergäbe, ergibt in C int. Es bringt zum Beispiel auch nichts, wenn man schreibt

```
realq = (long)(real*real)>>FLOG
```

Dann wird nämlich das im Register stehende Produkt (mit 32 gültigen Bit) durch EXT.L „erweitert“, danach sind die oberen 16 Bit verloren. Bei der effizienten Programmierung dieser Berechnungen hilft also nur der Inline-Assembler weiter.

Wird F so gewählt, daß bei den int-Variablen (Wertebereich $-2^{15} \dots +2^{15} - 1$) kein Überlauf auftreten kann, so ist dies auch für alle long-int-Variablen (Wertebereich -2^{31} bis $+2^{31} - 1$) gewährleistet. Die Beträge der long ints sind nämlich höchstens das Quadrat der größten int-Beträge (Bild 4) und

$$(-2^{15})^2 = 2^{30} < 2^{31} - 1$$

Ein int muß die rechten Seiten der Ungleichungen (4) und (5) aus Bild 4 aufnehmen können. Sie liegen sicher unter $8 \cdot F$, denn die Mandelbrotmenge erstreckt sich etwa im Bereich mit Realteil $-2 \dots 0,5$ und Imaginärteil $-1,2 \dots 1,2$. Für F wird also $2^{15} / 8 = 2^{12} = 4096$ gewählt.

Das Programm besteht aus zwei Teilen: Dem Hauptteil MANDNORM.C (Bild 5) sowie dem systemabhängigen Teil SYSTEM.H, der als Include-File eingebunden wird (für SW: Bild 6, für Farbe: Bild 7). SYSTEM.H muß sich bei der Compilation im gleichen Directory wie MANDNORM.C befinden.

Die Anpassung des Hauptteils auf andere Computer mit M68000 in C und Assembler oder auch in reinem Assembler ist relativ problemlos. Art und Syntax der Assembler-einbindung kann je nach C-Compiler unterschiedlich sein (Handbuch konsultieren). Die Punktsetz-Routine des Grafikteils liegt eventuell bereits als Bibliotheksfunktion vor.

Variationen

Grundsätzlich kann das Programm auch auf Rechner mit bloßem 16-Bit-Prozessor, etwa IBM-Kompatible, umgeschrieben werden, da hier die Multiplikation zweier 16-Bit-Werte in Maschinensprache ebenfalls einen 32-Bit-Wert ergibt. Einfach wird die Anpassung, wenn auch real und imag als 32-Bit Typen deklariert werden; damit kann auf Assembler völlig verzichtet werden. Die Geschwindigkeit ist dann natürlich nicht mehr so beeindruckend. Mit Hilfe des Programms MANDNORM ist ein „Zoom“ durch die Mandelbrotmenge in erträglicher Zeit möglich (Beispiele: Real-

```
/* MANDNORM.C: schnelles Berechnen der Mandelbrotmenge mittels normalisierter Zahlen. (als .TOS - Programm vereinbaren)
by Ch. Pahl, 1988 */

#include <stdio.h>
#include "system.h" /* systemabhängiger Teil, Bild 6 oder 7 */

#define F 4096 /* Normalisierungsfaktor */
#define FLOG 12 /* 2er-Logarithmus von F */
#define ITMAX 50 /* maximale Anzahl Iterationsschritte; grösserer Wert ergibt etwas genauere Darstellung */

/* der dargestellte komplexe Bereich, hier die gesamte Mandelbrotmenge: */
#define REALMIN (norm)(-2.58 * (float)F)
#define REALMAX (norm)( 1.17 * (float)F)
#define IMAGMIN (norm)(-1.172 * (float)F)
#define IMAGMAX (norm)( 1.172 * (float)F)

#define DELTAREAL ((REALMAX-REALMIN) / HRES) /* Differenz der Werte der Realteile 2er horizontal benachbarter Punkte */
#define DELTAIMAG ((IMAGMAX-IMAGMIN) / VRES) /* Differenz der Werte der Imaginärteile 2er vertikal benachbarter Punkte */

/* Die Division durch HRES bzw. VRES stellt die Division einer normalisierten Zahl durch eine nicht normalisierte dar, vgl. Text! */

main()
{
    norm real0, imag0; /* Startwerte */
    /* MEGAMAX-C stellt für Daten 4 Registervariablen zur Verfügung: */
    register norm real, imag; /* in der Iteration errechnete Werte */
    register supernorm realq, imagq; /* real^2, imag^2 */

    int h_kord, v_kord; /* Horizontal- bzw. Vertikalkoordinate des betrachteten Bildschirmpunktes */
    int farbe, itschritt;
    initgraphic();

    /* In zwei verschachtelten Schleifen wird nun jedem Punkt ein Realwert und ein Imaginärwert zugeordnet; die größten Imaginärwerte am oberen Bildschirmrand. Da die graphische Hochachse entgegengesetzt verläuft, läuft die zweite Schleife "rückwärts": */

    real0 = REALMIN;
    for( h_kord=0; h_kord<=(HRES-1); h_kord++ )
    {
        imag0 = IMAGMAX;
        for( v_kord=0; v_kord<=(VRES-1); v_kord++ )
        {
            real=real0; /* Iterationswerte initialisieren */
            imag=imag0;
            for( itschritt=0; itschritt<ITMAX; itschritt++ )
            {
                asm(
                    /* realq = real*real >> FLOG : */
                    move real, realq
                    muls realq, realq
                    move #FLOG, D0 /* D0 ist frei verfügbar */
                    asr.l D0, realq ;
                    /* imagq = imag*imag >> FLOG-1 : */
                    move imag, imagq
                    muls imagq, imagq
                    asr.l D0, imagq
                )
                if( realq+imagq > 4*F ) /* Abbruchkriterium */
                    break;
                asm(
                    /* imag = 2*real*imag + imag0 : */
                    muls real, imag
                    move #FLOG-1, D0
                    asr.l D0, imag
                    add imag0(A6), imag /* MEGAMAX-C benutzt A6 als Zeiger auf Stackbereich der lokalen Variablen */

                    /* real = realq - imagq + real0: (realq und imagq nach dem Abbruchkriterium im norm-Bereich, deshalb Word-Breite!) */
                    move realq, real
                    sub imagq, real
                    add real0(A6), real
                )
            }
            if( itschritt > ITMAX ) /* falls Punkt der Mandelbrotmenge an- */
                farbe=MANDFARB; /* gehört: vordefinierte Farbe */
            else /* andernfalls: */
                farbe = itschritt % FARBEN; /* Zuordnung einer Farbe */
            /* mittels modulo-Funktion */
            plot(h_kord,v_kord,farbe);

            imag0 -= DELTAIMAG;
        }
        real0 += DELTAREAL;
    }
    getch(); /* Programmende: RETURN-Taste */
}
```

Bild 5: Hauptteil des Programms „MANDNORM“

```

/* SYSTEM.H für MEGAMAX-C und SW-Monitor */
/* Festlegung compilerunabhängiger Typen: */
typedef int norm; /* für normalisierte Zahlen
mit kleinem Wertebereich, 16-Bit - Typ */
typedef long supernorm; /* für normalisierte Zahlen
aus grösserem Bereich, 32-Bit - Typ */

#define HRES 640 /* horizontale Auflösung */
#define VRES 400 /* vertikale Auflösung */
#define FARBEN 2 /* Anzahl Farben */
#define MANDFARB 1 /* Farbe der Mandelbrotmenge,
für Hardcopies besser 0 */
static unsigned *screenbase; /* Bildschirmanfanga-
adresse */

extern long xbios(); /* TOS-Funktion */

initgraphic()
{
/* xbios(2) liefert Bildschirmadresse: */
screenbase=(unsigned*)xbios(2);
/* ESC f ausgeben: Cursor unsichtbar: */
printf("\033f\n");
}

plot(x, y, farbe) /* Setzen eines Punktes */
register unsigned x, y;
int farbe;
{
if( x<HRES && y<VRES ) /* Rein vorsichtshalber:
eventuelle negative x, y werden
als zu grosse unsigned abgefangen */
if( farbe )
screenbase[y*40+x/16] |= 1 << 15-x%16;
/* else: weisse Punkte werden einfach
freigelassen */
}

```

Bild 6: Systemabhängiger Teil für SW-Monitor

Geänderte Konstanten:

```

#define F 16384
#define FLOG 14

#define REALMIN (supernorm)(-2.58 * (float)F )
#define REALMAX, IMAGMIN, IMAGMAX analog als supernorm:

```

Geänderte Deklarationen im Hauptprogramm:

```

supernorm real0, imag0;
register supernorm real, imag, realq, imagq;

```

Geänderte Iterationsschleife:

```

for( itschritt=0; itschritt<ITMAX; itschritt++)
{
if( real >= 2L*F || real< -2L*F
|| imag >= 2L*F || imag< -2L*F )
break; /* Vorwegnahme des Abbruch-
kriteriums */

asm(
move real, realq
muls realq, realq
move #FLOG, D0
asr.l D0, realq

move imag, imagq
muls imagq, imagq
asr.l D0, imagq
)
if( realq+imagq >= 4L*F )
break;

asm(
muls real, imag
move #FLOG-1, D0
asr.l D0, imag
add.l imag0(A6), imag
/* Hier und im Folgenden:
nun Long-Breite nötig! */

move.l realq, real
sub.l imagq, real
add.l real0(A6), real
)
}

```

Bild 9: Programmänderungen für höhere Auflösung

```

/* SYSTEM.H für MEGAMAX-C und Farbmonitor,
niedrige Auflösung; diese muß vor Programmstart
eingestellt werden */
/* Festlegung compilerunabhängiger Typen: */
typedef int norm; /* für normalisierte Zahlen
mit kleinem Wertebereich, 16-Bit - Typ */
typedef long supernorm; /* für normalisierte Zahlen
aus grösserem Bereich, 32-Bit - Typ */

#define HRES 320 /* horizontale Auflösung */
#define VRES 200 /* vertikale Auflösung */
#define FARBEN 16 /* Anzahl Farben:
wegen chaotischer Farbindizierung gewünschte Farb-
folge im Kontrollfeld so einstellen:
: 0: 1: 4: 3: 7: 9: 12: 11:
: 15: 2: 6: 5: 8: 10: 14: 13:
*/
#define MANDFARB 0 /* Farbe der Mandelbrotmenge */
static unsigned *screenbase; /* Bildschirmanfanga-
adresse */

extern long xbios(); /* TOS-Funktion */

initgraphic()
{
/* xbios(2) liefert Bildschirmadresse: */
screenbase=(unsigned*)xbios(2);
/* ESC f ausgeben: Cursor unsichtbar */
printf("\033f\n");
}

plot(x, y, farbe) /* Setzen eines Punktes */
register unsigned x, farbe;
{
register unsigned maske, wordnr;
register unsigned *wpointer;
if( x<HRES && y<VRES ) /* s. a. Bild 6 */
{
wpointer = screenbase + (y*80 + 4*(x/16));
maske = 1 << (15-x%16);
for( wordnr=0; wordnr<3; wordnr++)
{
asm( /* C hier umständlich! */
asr #1, farbe
bcs.s bit_set
:
*(wpointer++) &= ~maske; continue:
bit_set:
*(wpointer++) |= maske;
)
}
}
}

```

Bild 7: Systemabhängiger Teil für Farbmonitor

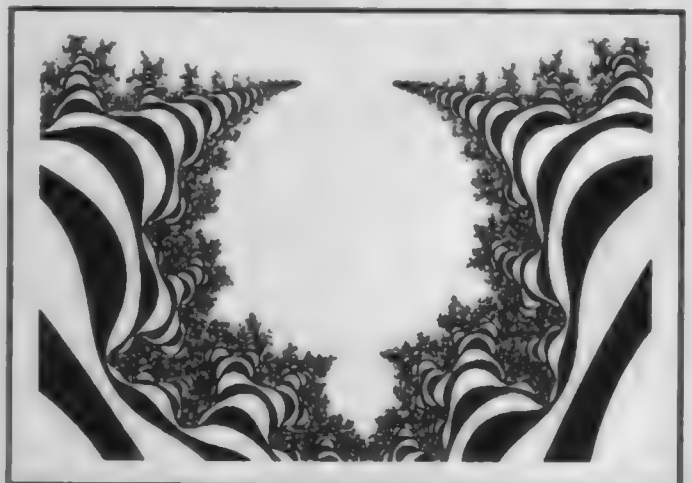


Bild 8: Die Mandelbrotmenge im Bereich Realteil -0.36...0.11, Imaginärteil -0.92...-0.62

teil $-0.32...-0.16$, Imaginärteil $0.77...0.87$ oder Realteil $-0.36...0.11$, Imaginärteil $-0.92...-0.62$, siehe Bild 8). Bei maximal 50 Iterationen benötigt das Programm zum Zeichnen der gesamten Mandelbrotmenge in Farbe ca. 50 Sekunden, in SW ca. 190 Sekunden, da viermal höhere Auflösung. Durch Ausnutzen sämtlicher Prozessorregister in Assembler (insbesondere in der Iterationsschleife) kann noch etwas höhere Geschwindigkeit erreicht werden. Auch auf dem SW-Schirm könnten unterschiedliche „Farben“ dargestellt werden, indem man ihn z.B. in 320×200 Quadrate zu je 4 Pixel aufteilt, dann stehen nämlich 5 Grauwerte zur Verfügung, weil in einem Quadrat 0 bis 4 Pixel gesetzt werden können. Die Genauigkeit kann natürlich nicht besser als $1/4096$ sein. Mit folgender Methode kann man die Auflösung jedoch ohne viel

zusätzliche Rechenzeit vervierfachen, also einen 16mal kleineren Bereich der Mandelbrotmenge noch darstellen: Auch real und imag werden als long int deklariert. Auf die Forderung, daß real und imag im int-Wertebereich liegen, kann man verzichten, wenn man diese Variablen vor Berechnen von realq und imagq auf Überschreiten des Wertes $2 \cdot F$ überprüft. Fällt der Test positiv aus, wird die Iteration abgebrochen, da das Abbruchkriterium ja sicher erfüllt wäre. F muß also nur noch so gewählt werden, daß $2 \cdot F$ im Integerbereich liegt. Wählt man F so, daß nur $2 \cdot F - 1$ noch in diesem Bereich liegt und bricht dann schon bei Erreichen von $2 \cdot F$ ab, macht man praktisch keinen Fehler, da der Mikroprozessor sowieso stets abrundet. Jetzt kann für F sogar $2^{15} / 2 = 16384$ gewählt werden. Die Änderungen am Programm zeigt Bild 9.

Apropos Abrunden: Der Grund für kleine Asymmetrien in der Grafik ist, daß (bei der Bitverschiebung) positive und negative Zahlen beide abgerundet werden; vollständige Symmetrie erreicht man leicht, wenn man der Variablen imag0 vor Beginn der Iteration den Absolutwert von $k \cdot \text{imag}_0 \cdot F$ zuweist.

Eine weitere Anwendung normalisierter Zahlen finden Sie in [1].

Literatur

- [1] Obermaier, Anton: Sin und cosin mit Integerarithmetik, mc 6/87
- [2] Mandelbrot, B. B.: Fractals: Form, Chance and Dimension, Freeman, San Francisco 1977

Zusammen mit der neuen Autofokus-Spiegelreflexkamera Dynax 7000i stellte Minolta Chipkarten vor, die per Software für spezielle Anwendungen die Kamerafunktionen gezielt steuern. Neun Karten sind mit festen Programmen versehen. Eine frei programmierbare erlaubt den Zugriff auf einige ausgewählte Funktionen: die Fokussierstoptaste kann zum Umschalten auf das Autofokuszielfeld oder den kontinuierlichen Autofokus benutzt werden, die akustische Warnung vor zu langen Verschlusszeiten kann abgeschaltet werden, die Bildzahl kann statt hoch- heruntergezählt werden, so daß immer die verbleibende Bildzahl angezeigt wird, die automatische Rückspulung am Filmende kann abgeschaltet werden (sie erfolgt dann erst bei Betätigung der Rückspulstarttaste), der Rücktransport kann vorzeitig gestoppt werden, damit die Filmzunge noch aus der Patrone herauschaut, die Zahl der einstellbaren Belichtungsfunktionen kann eingeschränkt werden.

Die neun festprogrammierten Karten sind jede auf eine bestimmte Spezialanwendung oder Funktionserweiterung eingestellt:

- Belichtungsreihen in wählbaren Schritten von drei, fünf oder sieben Aufnahmen in kurzer Folge
- Programmshift für drei Aufnahmen in schneller Folge bei jeweils gleicher Belichtung, die erste bei normaler Verschlusszeit, die zweite bei kürzerer und die dritte bei längerer Verschlusszeit. Die

Minolta mit Chipkarten



Vorbereitung für eine spezielle Anwendung

Zeitunterschiede sind in drei Stufen einstellbar.

- für Sport- oder Action-Aufnahmen eine möglichst kurze Belichtungszeit unter Berücksichtigung der Objektivbrennweite und des Objektstandes
- Speicherung aller fototechnischen Daten eines Filmes
- automatische Lichter-/Schatten-Bewertung
- Phantasie-Effekt durch Verstellen der Schärfe während der Belichtung; bei großer Blende entsteht ein Weichzeichner-, bei kleiner ein Zoomeffekt

- für Portraits optimale Schärfentiefe bei Verschlusszeiten aus freier Hand
- für Nahaufnahmen optimale Schärfentiefe
- möglichst große Schärfentiefe

Zur Anwendung einer Chip-Karte öffnet man einfach eine Klappe an der Seite der Kamera und steckt sie in ein Fach (siehe Bild). Jede Karte kostet circa 50 DM, die Dynax 7000i (ohne Karten, mit Zoom 28-80 mm) circa 1200 DM. Die Kamera ist auch ohne Karten voll funktionsfähig.

Die Autofokuseinrichtung soll doppelt so schnell arbeiten wie die des Vorgängermodells 7000; auch in der Einfachheit der Bedienung will man einen großen Schritt nach vorn getan haben. Wichtige Tasten sind leicht erreichbar und übersichtlich angeordnet; weniger benötigte wurden zum Beispiel an der Innenseite der Kartenfachklappe verborgen.

Bemerkenswert ist die sogenannte Prädiktions-Steuerung des Autofokus-Systems, eine Art dynamische Nachführung der Entfernungseinstellung: Sie stellt fest, ob sich das Objekt zur Kamera hin oder von ihr weg bewegt, und errechnet die Bewegungsgeschwindigkeit. Die Schärfe wird auch dann noch nachjustiert, wenn der Spiegel hochklappt, bis zu dem Moment, in dem sich der Verschluss öffnet. Selbst bei äußerst schwachem Licht weiß sich die Kamera zu helfen: Dann nämlich wird ein Hilfsstrahl ausgesandt, an dem die Autofokus-Steuerung noch in neun Metern Entfernung Objekte scharf stellen kann. br

Robert Tolksdorf

Schreibschutz für die Atari-Festplatte

Das Programm (Bild 1) installiert sich als Accessory und ist durch Anklicken von „Schreibschutz“ im „Desk“-Menü steuerbar. Beim Aufruf erscheint eine Dialogbox wie in Bild 2. In ihr läßt sich festlegen, welche der Harddisk-Partitions schreibgeschützt werden sollen.

Der angeschaltete Schreibschutz wird in der Dialogbox durch ein Häkchen dargestellt. Ein Klick auf den Namen der gewünschten Partition schaltet den Schutz ein oder aus. Nicht installierte logische Laufwerke – im abgebildeten Fall die Einheit F – werden hell angezeigt und sind nicht wählbar. Mit „ABBRUCH“ bleibt alles unverändert, falls das Accessory aus Versehen angeklickt wurde. Das Programm gibt intern eine entsprechende Fehlermeldung weiter, wenn auf eine schreibgeschützte Partition zugegriffen werden soll. Allerdings können einige Programme mit ihr nichts anfangen, da der Schreibschutz von Disketten BIOS-intern mit der bekannten Alarmbox behandelt wird, und somit dieser Fehler gar nicht auftauchen sollte. Wenn ein Programm also auf geschützte logische Laufwerke schreiben will, und so tut, als ob alles richtig gelaufen wäre, liegt es nicht an HDLOCK. Prominentestes Beispiel solcher etwas schlampiger Programmierung ist der GEM-Desktop. HDLOCK kann übrigens auch mit einer RAM-Disk arbeiten. Wenn der Treiber sich korrekt installiert hat, fängt HDLOCK auch solche Schreibzugriffe ab.

Das Programm

HDLOCK wurde mit TDI Modula-2 geschrieben. Die Resource für die Dialogbox ist mit der „Prozedur“ RESOURCEDATA0 direkt in den Programmcode eingebettet. Diese Daten und der entsprechende Initialisierungsteil in den Zeilen 102 bis 112 wurden mit dem Utility RSCMAKER erzeugt. Diese Vorgehensweise spart Speicherplatz ein, da GEM bei einem Resource-Load-Aufruf aus einem Accessory mehrere Kilobyte unsinnig verschwendet. Bei der Auswertung der Benutzereingabe in der Dialogbox

Wer – vielleicht aus Furcht vor den inzwischen weitverbreiteten Virus-Programmen – Schreibzugriffe auf die Festplatte (oder auf eine RAM-Disk) verhindern will, kann das mit dem hier vorgestellten Accessory HDLOCK tun. Ein sehr nützliches Utility, das in Modula-2 geschrieben wurde.

werden in locked Flags für den Schreibschutz gesetzt (Zeilen 139-145). Über diesen Bitvektor wird dann entschieden, ob eine logische Diskstation schreibgeschützt ist. Dazu wird die Prozedur MyRWAbs als neuer Laufwerkstreiber dem Betriebssystem durch Verän-

```

1: MODULE HDLock ;
2:
3: (* Robert Tolksdorf, 15.2.88 *)
4: (* $T-$)($S-$)($A-$) (* Optimieren *)
5:
6: FROM SYSTEM IMPORT ADDRESS, ADR, SETREG, CODE;
7: FROM BIOS IMPORT DriveMap;
8: FROM GEMDOS IMPORT EWrPro, EUnDev;
9: FROM XBIOS IMPORT SuperExec;
10: FROM GENAESbase IMPORT Object, AccessoryOpen, FormStart,
11: FormFinish, FormGrow, FormShrink,
12: Normal, Checked, Disabled;
13: FROM AESApplications IMPORT ApplInitialise;
14: FROM AESEvents IMPORT EventMessage;
15: FROM AESForms IMPORT FormDialogue, FormDo, FormCenter;
16: FROM AESMenus IMPORT MenuRegister;
17: FROM AESObjects IMPORT ObjectDraw, ObjectChange;
18: FROM AESResources IMPORT ResourceObjectFix;
19:
20: CONST DIALOG = 0 ;
21: DRIVEC = 1 ;
22: DRIVED = 2 ;
23: DRIVEE = 3 ;
24: DRIVEF = 4 ;
25: ABBRUCH = 8 ;
26:
27: TYPE rwProcType = PROCEDURE (CARDINAL, CARDINAL, CARDINAL, ADDRESS, CARDINAL);
28:
29: VAR RWVector [0476H] : ADDRESS; (* Vektor *)
30: OldRWVector : rwProcType;
31: MapOfDrives [04C4H] : BITSET; (* angeschlossene Laufwerke *)
32: locked : BITSET; (* geschützte Laufwerke *)
33: DrvMap : ARRAY [DRIVEC..DRIVEF] OF CARDINAL;
34: (* Variablen für Umgang mit AES *)
35: msgbuff : ARRAY [0..7] OF CARDINAL;
36: applID, accID, x, y, w, h, result : INTEGER;
37: (* Variablen zum Initialisieren der Resource *)
38: i : CARDINAL;
39: TreeAddr : POINTER TO ARRAY [0..0] OF ADDRESS;
40: ObjectAddr : POINTER TO ARRAY [0..8] OF Object;
41:
42: (* Resourcedaten direkt in den Programmcode eingebettet *)
43: (* $P-$ *)
44: PROCEDURE RESOURCEDATA0;
45: BEGIN
46: CODE(00000H, 0009AH, 0009AH, 0009AH, 0009AH, 0009AH, 00024H, 0009AH);
47: CODE(0009AH, 00172H, 00009H, 00001H, 00000H, 00000H, 00000H, 00000H);
48: CODE(00000H, 00176H, 04300H, 04400H, 04500H, 04600H, 04844H, 04C6FH);
49: CODE(0636BH, 02062H, 07920H, 0526FH, 06265H, 07274H, 02054H, 06F6CH);
50: CODE(06873H, 0646FH, 07266H, 02C20H, 03139H, 03838H, 00053H, 06368H);
51: CODE(07265H, 06962H, 07363H, 06875H, 0747AH, 02084H, 06B64H, 06572H);
52: CODE(06E20H, 06681H, 07220H, 04C61H, 07566H, 07765H, 0726BH, 0203AH);
53: CODE(00028H, 00820H, 06B65H, 06E6EH, 07A65H, 06963H, 0686EH, 06574H);
54: CODE(02061H, 06B74H, 06976H, 0656EH, 02053H, 06368H, 07574H, 07A29H);
55: CODE(00041H, 04242H, 05255H, 04348H, 00000H, 0FFFFH, 00001H, 00008H);
56: CODE(00014H, 00000H, 00010H, 00002H, 01100H, 00000H, 00000H, 00028H);
57: CODE(00008H, 00002H, 0FFFFH, 0FFFFH, 0001AH, 00005H, 00008H, 00000H);

```

Bild 1. Ein Modula-2-Programm schützt die Festplatte

```

58: CODE(00024H,00002H,00006H,00005H,00001H,00003H,0FFFFH,0FFFFH);
59: CODE(0001AH,00005H,00008H,00000H,00026H,00008H,00006H,00005H);
60: CODE(00001H,00004H,0FFFFH,0FFFFH,0001AH,00005H,00008H,00000H);
61: CODE(00028H,0000EH,00006H,00005H,00001H,00005H,0FFFFH,0FFFFH);
62: CODE(0001AH,00005H,00008H,00000H,0002AH,00014H,00006H,00005H);
63: CODE(00001H,00006H,0FFFFH,0FFFFH,0001CH,00000H,00000H,00000H);
64: CODE(0002CH,00004H,00001H,00020H,00001H,00007H,0FFFFH,0FFFFH);
65: CODE(0001CH,00000H,00000H,00000H,0004DH,00002H,00003H,00023H);
66: CODE(00001H,00008H,0FFFFH,0FFFFH,0001CH,00000H,00000H,00000H);
67: CODE(00071H,00004H,00004H,0001FH,00001H,00000H,0FFFFH,0FFFFH);
68: CODE(0001AH,00027H,00000H,00000H,00091H,0001DH,00006H,00008H);
69: CODE(00001H,00000H,0009AH);
70: END RESOURCEDATA0;
71:
72: (* Die neue Routine mit Überprüfung auf Schreibschutz *)
73: PROCEDURE MyRWabs(dev,recno,count:CARDINAL; buf:ADDRESS; rwflag:CARDINAL);
74: BEGIN
75: CODE(048E7H,07FFCH); (* movem D1..A5,-(A7) *)
76: IF dev IN MapOfDrives THEN
77: IF ((rwflag=1) OR (rwflag=3)) AND (dev IN locked) THEN
78: SETREG(0,EWRProc); (* Write protected Fehler in D0 *)
79: ELSE (* kein geschützter Schreibzugriff -> normal ausführen *)
80: OldRWVector(dev,recno,count,buf,rwflag);
81: END
82: EWRProc
83: SETREG(0,EUnDev); (* Unknown device in D0 *)
84: END;
85: CODE(04CDFH,03FFE); (* movem (A7)+,D1..A5 *)
86: END MyRWabs;
87:
88: (*$P- Routine wird im Supervisor-Mode gerufen *)
89: PROCEDURE InstallVectors;
90: (* neue Routine installieren und alte vermerken *)
91: BEGIN
92: OldRWVector := rwProcType(RWVector);
93: RWVector := ADDRESS(MyRWabs);
94: CODE(04E75H); (* rts *)
95: END InstallVectors;
96:
97: BEGIN
98: (* Accessory anmelden und installieren *)
99: applID:=ApplInitialise();
100: accID:=MenuRegister(applID,' Schreibschutz');
101: (* Initialisierung der Resource - erzeugt mit RSCMaker *)
102: TreeAddr := ADDRESS(RESOURCEDATA0) + 370;
103: TreeAddr[0] := TreeAddr[0] + ADDRESS(RESOURCEDATA0);
104: ObjectAddr := ADDRESS(RESOURCEDATA0) + 154;
105: FOR i := 0 TO 8 DO
106: WITH ObjectAddr[i] DO
107: IF (type#20) & (type#24) & (type#25) & (type#27) THEN
108: spec := spec + ADDRESS(RESOURCEDATA0);
109: END;
110: ResourceObjectFix(ObjectAddr,i); (*0.10a*)
111: END;
112: END;
113: FormCenter(ObjectAddr,x,y,w,h);
114: (* Buttons entsprechend angeschlossenen Laufwerken enablen *)
115: DrvMap[DRIVEC]:=2; DrvMap[DRIVED]:=3;
116: DrvMap[DRIVEE]:=4; DrvMap[DRIVEF]:=5;
117: FOR i:=DRIVEC TO DRIVEF DO
118: IF DrvMap[i] IN DriveMap() THEN
119: ObjectChange(ObjectAddr,i,0,x,y,w,h,Normal,0);
120: END;
121: END;
122: locked:=1; (* keine Laufwerke geschützt *)
123: (* Routine installieren *)
124: SuperExec(PROC(InstallVectors));
125: LOOP
126: (* Auf Accessory-Aufruf warten *)
127: EventMessage(ADR(msgbuff));
128: IF (msgbuff[0]=AccessoryOpen) THEN
129: (* Dialog anzeigen und ausführen *)
130: FormDialog(FormStart,x,y,w,h,x,y,w,h);
131: FormDialog(FormGrow,0,0,0,0,x,y,w,h);
132: ObjectDraw(ObjectAddr,DIALOG,99,x,y,w,h);
133: result:=FormDo(ObjectAddr,-1);
134: (* Dialog abschließen *)
135: FormDialog(FormShrink,0,0,0,0,x,y,w,h);
136: FormDialog(FormFinish,x,y,w,h,x,y,w,h);
137: ObjectChange(ObjectAddr,result,0,x,y,w,h,Normal,0);
138: (* Ausgewähltes Laufwerk umschalten *)
139: IF (result>=DRIVEC) AND (result<=DRIVEF) THEN
140: IF DrvMap[result] IN locked THEN
141: EXCL(locked,DrvMap[result]);
142: ELSE
143: ObjectChange(ObjectAddr,result,0,x,y,w,h,Checked,0);
144: INCL(locked,DrvMap[result]);
145: END;
146: END;
147: END;
148: END;
149: END HDLock.

```

HDLock by Robert Tolksdorf, 1988

Schreibschutz ändern für Laufwerk :
(/ kennzeichnet aktiven Schutz)

☒ C ☒ D ☐ E ☐ F

Bild 2. So sieht die Dialogbox zum Programm HDLock aus

dem des Vektors bei \$0476 bekanntgemacht. Die Routine InstallVectors muß dabei im Supervisor-Modus ablaufen, da sie auf geschützten RAM-Bereichen arbeitet. Bei einem Laufwerkszugriff wird dann zunächst geprüft, ob ein durch dev identifiziertes Gerät überhaupt existiert. Ist dies der Fall, und zeigt flag einen Schreibzugriff an, wird mit locked abgefragt, ob Schreiben erlaubt sein soll. Wenn nicht, gibt MyRWabs im Register D0 den Fehlercode für schreibgeschützt zurück. Ansonsten reicht die Routine den Aufruf einfach an den eigentlichen Harddisk-Treiber weiter. Falls das Gerät nicht angemeldet ist, wird der Fehler Unknown Device gemeldet. Die Compileroption P bei InstallVectors und der Assemblercode RTS überwinden den normalen Modula-Aufrufmechanismus, der bei einer Routine im Supervisor-Modus einen Stack-Fehler erzeugen würde. In MyRWabs werden durch zwei mit CODE eingefügte Assemblerstatements sicherheitshalber die Register D1...D7 und A1...A5 auf den Systemstack gerettet und wiederhergestellt. Die Allokierung von Variablen an feste Adressen und der Prozedurtyp erlauben eine elegante und übersichtliche Formulierung von systemnahen Funktionen – sicherlich eine der Stärken von Modula-2.

Compilieren

Beim Compilieren mit dem TDI-Modula-2 Übersetzer müssen Sie beim Linken die Option „QUERY“ im Modula-Options Accessory setzen und bei der Frage nach „GEMX.LNK“ die Datei „GEMACCX.LNK“ auswählen. Dadurch wird ein eingeschränktes Laufzeitsystem hinzugebunden, das weniger Speicherplatz belegt. Noch kürzer wird das Accessory, wenn Sie die Linker-Option „OPT“ setzen. Abschließend müssen Sie noch die Extension des erzeugten Programms auf „.ACC“ ändern und das Programm auf die Boot-Partition kopieren.

Literatur

- [1] Modula-2/ST User's Manual, TDI Software LTD
- [2] Brückmann, Englisch, Gerits: Atari ST Intern, Data Becker
- [3] Krämer, Riebl, Hübner: Das TOS-Listing, Band 1, Heise Verlag

Eckart Winkler

Typisch Prolog

Differenzieren und Vereinfachen symbolischer Ausdrücke

In Prolog existieren zwei verschiedene Operatoren, die als Zuweisungen aufzufassen sind. Betrachten wir hierzu einen Ausdruck wie $3*4+5$. Eine Zuweisung an die Variable X würde in den meisten Sprachen zur Folge haben, daß der Wert 17 in die Speicherstelle geschrieben wird, die X entspricht. In Prolog müssen wir unterscheiden. „ $X=3*4+5$ “ setzt die Variable auf den Ausdruck $3*4+5$. Dieser Ausdruck hat zwar den Wert 17, von der Struktur her betrachtet ist er jedoch ungleich 17. Wollen wir erreichen, daß X auf 17 gesetzt wird, müssen wir eingeben „ X is $3*4+5$ “.

Der Operator „is“ zwingt den Prolog-Interpreter also zur Auswertung, während „=“ den Ausdruck unverändert läßt. Eine Auswertung mittels „is“ ist natürlich unmöglich, wenn im Ausdruck Variablen oder Text-Atome auftreten. Demgegenüber ist „=“ in der Lage, in gewissen Fällen eine Zuordnung an die vorkommenden Variablen zu treffen. Der Interpreter versucht jedenfalls immer, die auftretenden Variablen so zu belegen, daß die Gleichheit der Ausdrücke gilt.

Prolog kann zum Beispiel bei der Eingabe von $X*b+3=(a+1)*b+Y$ die Variablen X und Y belegen, nämlich mit $X=a+1$ und $Y=3$. Bei dieser Belegung sind beide Ausdrücke identisch. Man sagt dabei, die Ausdrücke werden unifiziert. Die Unifikation klappt nicht bei Eingabe von $X+3=Y*3$. (An dieser Stelle bleibt noch anzumerken, daß Variablen stets mit einem Großbuchstaben beginnen, Text-Atome, die formell den Zahlen gleichzustellen sind, hingegen mit einem Kleinbuchstaben. $X1$ und xyz sind also Variablen, $a1$ und abc jedoch Text-Atome). Dieser Mechanismus der Unifikation kann leicht dazu verwendet werden, mit Ausdrücken zu operieren. Beispielsweise kann man an ein mehrfaches Produkt noch einen Faktor anhängen oder von einer mehrfachen Summe einen Summanden abspalten: $X+Y=a+b+c+d$ ergibt die Zuordnungen $X=a+b+c$ und $Y=d$. Warum beispielsweise nicht $X=a$, $Y=b+c+d$ gesetzt wurde, hängt mit der Art und Weise zusammen, wie der Operator $+$ definiert ist. Denn es gilt das Assoziativgesetz, und Summen werden von

Was die Programmiersprachen der Künstlichen Intelligenz gegenüber herkömmlichen Sprachen vor allem auszeichnet, ist die problemlose Behandlung von Listen und Bäumen. Prolog geht sogar noch einen Schritt weiter und ermöglicht das Rechnen mit symbolischen Rechenausdrücken.

links nach rechts ausgewertet. Diese beiden Eigenschaften sowie eine Zahl, die den Vorrang gegenüber anderen Operatoren angibt, bestimmen das Verhalten des Operators.

Operatoren Marke Eigenbau

Prolog gestattet es, eigene Operatoren zu definieren. Damit kann zum Beispiel eine im System nicht vorhandene Exponentiation eingeführt werden. Hierzu verwenden wir wie in Basic das Zeichen $^$. Allgemein wird die Exponentiation von rechts nach links ausgewertet, das wird in Prolog durch das Symbol xy gekennzeichnet. Auch hat $^$ normalerweise Vorrang vor $+$ und $*$. Um dies festzulegen, müssen wir $^$ einen niedrigeren Index zuordnen, als $+$ und $*$ besitzen. Wir wählen jetzt einmal 350.

Dann hat die Anweisung zur Definition als Operator die Form $op(350,xy,')$. Die Zahl

350 bedarf eventuell einer Anpassung; bei bereits vorhandenem Operator $^$ kann die entsprechende Zeile im Listing von Bild 1 ganz wegfallen. Wir wollen mit dem Operator nur symbolisch umgehen. Es ist also gar nicht erforderlich, daß wir außerdem noch definieren, wie derartige Potenzen ausgewertet

werden. Weil man so einfach mit Rechenausdrücken umgehen kann, ist Prolog wie geschaffen für Programme, die Funktionen ableiten (differenzieren). Es gibt genügend Gründe, sich mit diesem Problem zu befassen. Die Ableitung einer Funktion gibt ja die Steigung des Funktionsgraphen in jedem Punkt an. In der Physik läßt sich die Ableitung jedes Weg-Zeit-Gesetzes als Geschwindigkeits-Zeit-Gesetz deuten, die nochmalige Ableitung als Beschleunigungs-Zeit-Gesetz.

Die Ableitungsregeln wollen wir nun in Prolog formulieren. Zunächst zur Schreibweise: Die Ableitung einer Funktion f wird oft mit f' bezeichnet. Dabei muß die Differentiationsvariable jedoch klar sein. In Prolog müssen wir das etwas anders machen. Hier werden wir ein Prädikat „dif“ definieren („dif“ steht für „differenziere“). Es soll die Form $dif(U,X,V)$ haben. Hierbei ist U der zu differenzierende Ausdruck, X die

```
:-op(350,xfy,').
% Operator ^

in(X,X):-!.
in(X,U):-U=..[_ ,V,W],(in(X,V);in(X,W)).
in(X,U):-U=..[_ ,V],in(X,V).

dif(U,X,V):-numbertvars(U,0,Z),Z>0,!,fail.
% Keine Variablen
% Elementare Funktionen
dif(sin(X),X,cos(X)):-!.
dif(cos(X),X,-sin(X)):-!.
dif(exp(X),X,exp(X)):-!.
dif(ln(X),X,1/X):-!.
dif(X^N,X,R):-not in(X,N),R=N*X^(N-1),!.
dif(X,X,1):-!.
dif(C,X,0):-atomic(C),!.
dif(U,X,V):-U=..[F,X],name(F,L),append(L,[33],M),
name(G,M),V=..[G,X],!.
% f(x)' = fS(x)
% Umwandlungen
dif(U^V,X,R):-dif(exp(V*ln(U)),X,R),!.
dif(U/V,X,R):-dif(U*v^(-1),X,R),!.
dif(-U,X,R):-dif(U,X,A),R=-A,!.
% Differentiationsregeln
dif(U+V,X,R):-dif(U,X,A),dif(V,X,B),R=A+B,!.
dif(U-V,X,R):-dif(U,X,A),dif(V,X,B),R=A-B,!.
dif(U*V,X,R):-dif(U,X,A),dif(V,X,B),R=A*V+U*B,!.
dif(S,X,R):-S=..[F,U],dif(S,U,A),dif(U,X,B),R=A*B,!.
% Kettenregel

dif(U,X,1,D):-dif(U,X,D),!.
dif(U,X,N,D):-dif(U,X,B),M is N - 1,dif(S,X,M,D).
```

Bild 1. Das Prolog-Programm zur Ableitung von Funktionen

Variable, nach der abgeleitet werden soll, und V das Resultat. Ein typischer Aufruf wäre also $\text{diff}(x^2, x, D)$.

Wir wollen nun Schritt für Schritt nachvollziehen, wie das fertige Programm von Bild 1 entsteht. Zunächst soll das System mit vier elementaren Funktionen versorgt werden: Sinus, Cosinus, e^x und der natürliche Logarithmus (e ist natürlich die Eulersche Zahl, und sie hat den Wert $e=2.718281828$). e^x wird oft mit „exp(x)“ bezeichnet, dies tun wir auch. e ist gleichzeitig die Basis des natürlichen Logarithmus ln.

Zu allen genannten Funktionen ist die Ableitung bekannt. Die Ableitung von $\sin(x)$ beispielsweise ist $\cos(x)$. Die Regel hierfür muß lauten: $\text{diff}(\sin(x), x, \cos(x))$. In Worten heißt das: Leiten wir $\sin(x)$ nach x ab, lautet das Resultat $\cos(x)$. Der Cut mit ! in dieser Regel schließt ein Backtracking aus. Wenn der Benutzer eine zusätzliche Lösung suchen läßt, wird ihm mitgeteilt, daß eine solche nicht existiert. Dies ist ja auch richtig.

Leicht zu bestimmen ist auch die Ableitung von Ausdrücken der Form x^n . Die Ableitung lautet $n \cdot x^{(n-1)}$. Hier muß man jedoch vorsichtig sein: das ist nur richtig, wenn n von x unabhängig ist. Bevor wir diese Regel also nach Prolog übertragen können, müssen wir testen, ob x in n vorkommt. Wenn ja, ist die Ableitungsregel nicht anwendbar.

Für diesen Test schreiben wir uns das Prädikat „in“. $\text{in}(X, U)$ soll wahr sein, wenn X im Ausdruck U vorkommt. Da der Ausdruck beliebig verschachtelt sein kann, hilft nur rekursive Formulierung.

Zerlegen in eine Liste

Hierzu benötigen wir den Operator $=..$. Er liefert zu jedem Ausdruck eine Liste, deren erstes Element der Operator mit der geringsten Priorität ist. Die weiteren Elemente sind dessen Operanden.

Im Fall der Rechenoperatoren wird die gelieferte Liste also immer drei Argumente haben. Beispiele:

$3+4=..L$ ergibt

$L=[+, 3, 4], a+b+3/\cos(x)=..L$ ergibt

$L=[+, a+b, 3/\cos(x)]$.

Weil auch Funktionen wie cos und exp auftreten können, müssen wir jedoch auch mit zweielementigen Listen rechnen:

$\cos(3+4)=..L$ liefert $L=[\cos, 3+4]$.

Mit Hilfe von $=..$ können wir unseren Ausdruck also zerlegen. Welcher Operator auftritt, ist unerheblich. Deshalb können wir hierfür die anonyme Variable verwenden. Es kommt nur auf die Operanden an. Auf die wird das Prädikat wieder angewandt.

Bei den Rechen-Operatoren genügt es, wenn X in einem von zwei Operanden auftritt. Als Abbruchbedingung ist der Fall angegeben, daß der Ausdruck gleich dem zu testenden Element ist. Dieser Fall wird immer erreicht, da bis auf die unterste Stufe zerlegt wird. Die Realisierung von „in“ ist ebenfalls in Bild 1 angegeben.

Damit können wir unsere nächste Ableitungsregel formulieren. Mit „not in(X,N)“ wird gewährleistet, daß X nicht in N vorkommt. Dahinter wird das Resultat auf den bereits erwähnten Ausdruck gesetzt: $R=N \cdot X^{(N-1)}$. Sonderfälle dieser Ableitungsregel sind die Ausdrücke der Form X und C. X nach X abgeleitet, ergibt 1. C nach X abgeleitet, ergibt 0, wenn C atomic ist, also Zahl oder Text-Atom.

Es folgen nun einige Regeln, die keine Differentiationsregeln im eigentlichen Sinne sind. Hier werden Ausdrücke umgeformt, d. h. eine Form hergestellt, die von den übrigen Regeln behandelt werden kann. Beispielsweise existiert keine Regel, um eine beliebige Potenz U^V abzuleiten. Es ist aber bekannt, daß wir für U^V auch $\exp(V \cdot \ln(U))$ schreiben können. Wir müssen im Fall U^V also dif erneut aufrufen, und zwar für $\exp(V \cdot \ln(U))$.

Die Regeln für exp und ln kennen wir bereits. Das Produkt wird von der Produktregel erledigt, die wir noch besprechen werden. Somit können wir U^V auf andere bekannte Regeln zurückführen. Genauso kann man U/V behandeln. Dieser Fall läßt sich kürzer lösen als mit der Quotientenregel. Wir formen U/V einfach in $U \cdot V^{-1}$ um. Für das Produkt haben wir dann wieder die Produktregel, die Potenz wird durch die Regel von X^N abgedeckt.

Die letzte dieser Umformungsregeln behandelt ein negatives Vorzeichen. Dieses kann bei der Differentiation weggelassen und später wieder hinzugefügt werden.

Ableitungsregeln

Nun kommen wir zu den Regeln, die als Ableitungsregeln aus der Schule bekannt sind. Am einfachsten zu merken sind die Additions- und die Subtraktionsregel

$(u+v)'=u'+v'$ und

$(u-v)'=u'-v'$.

Wenn wir also eine Summe ableiten wollen, müssen wir die Summanden einzeln ableiten und die Resultate wieder addieren. Genau dies geschieht bei der entsprechenden Prolog-Regel (analog für die Subtraktion).

Etwas komplizierter verhält es sich für ein Produkt. Die Regel hierfür besagt: $(u \cdot v)'=u' \cdot v + u \cdot v'$. Aber auch dies kann direkt in einer Prolog-Regel ausgedrückt wer-

den. Wir berechnen zunächst die Ableitungen von u und v einzeln und bilden hiermit den erforderlichen Ausdruck. Wie bereits erwähnt, ist die Regel für Quotienten u/v nicht explizit programmiert, sondern wird durch Umwandlung in $u \cdot v^{-1}$ auf die Produktregel zurückgeführt.

Knifflig wird es nun bei der Kettenregel. Sie kommt zur Anwendung, wenn zwei Funktionen verschachtelt vorliegen, allgemein $f(g(x))$. Dann müssen wir so tun, als hätten wir $f(x)$ gegeben. Dies können wir ableiten und erhalten $f'(x)$. Für x setzen wir dann jedoch wieder $g(x)$ ein und haben $f'(g(x))$. Schließlich leiten wir noch $g(x)$ nach x ab und multiplizieren dies mit $f'(g(x))$. Dies ist das Resultat: $f'(g(x)) \cdot g'(x)$.

Ein Beispiel: Gesucht ist die Ableitung von $\sin(\exp(x))$. Dann leiten wir zunächst $\sin(x)$ ab, das ist $\cos(x)$. Einsetzen von $\exp(x)$ für x ergibt: $\cos(\exp(x))$. Dies multiplizieren wir mit der Ableitung von $\exp(x)$, das ist $\exp(x)$ selbst. Das Ergebnis ist also $\cos(\exp(x)) \cdot \exp(x)$.

Jetzt müssen wir die Kettenregel nur noch in Prolog formulieren. Zunächst gilt es, den gegebenen Ausdruck S als „Schachtelausdruck“ zu identifizieren. Dazu dient uns wie beim Prädikat „in“ der Operator $=..$. In der Form $S=..[F, U, _]$ liefert er uns in F den Namen der Funktion und in U das erste Argument der Funktion. Die weiteren Argumente sind nicht von Interesse.

Wir müssen nun U in S durch X ersetzen und nach X ableiten, schließlich wieder X durch U ersetzen. Dies geht in Prolog in einem Schritt. Hier können wir S nach U ableiten, auch wenn U seinerseits ein komplizierter Ausdruck ist. Weiter ist die Ableitung von U nach X erforderlich. Beide Resultate werden miteinander multipliziert.

Die Kettenregel kommt übrigens auch bei Potenzen zur Anwendung, wenn nämlich die Basis nicht x, sondern ein anderer Ausdruck ist. Dies ist zum Beispiel bei $(x-1)^3$ der Fall.

Unbekannte Funktionen und Mehrfachableitungen

Was passiert eigentlich, wenn eine dem System unbekannte Funktion, zum Beispiel der Tangens $\tan(x)$, eingegeben wird? Aus Gründen der Vollständigkeit wollen wir dann an den Funktionsnamen ein großes S anhängen, um die Ableitung zu kennzeichnen. Dies ist keine Spielerei, sondern wird auch zu praktisch verwertbaren Ergebnissen führen.

Wir müssen zunächst den Funktionsnamen isolieren. Dies geht wieder mit $U=..[F, X]$. Unser System ist nur für Funktionen mit einer Variablen gedacht, daher hat die Liste

nur zwei Argumente. Nun zerlegen wir den Namen F mit name in seine Bestandteile. name liefert eine Liste mit den ASCII-Werten der Zeichen von F. name(tan,L) ergibt zum Beispiel L=[116,97,110]. An diese Liste können wir mit append leicht den ASCII-Wert von S, nämlich 83, anhängen, und dann gehen wir den gesamten Weg wieder rückwärts.

Die im Programm erste Regel für dif ist bisher unerwähnt geblieben. Sie sorgt dafür, daß in den eingegebenen Ausdrücken keine Variablen auftreten. Damit könnte es nämlich zu Endlos-Schleifen kommen. Grundlegend hierfür ist das Prädikat numbervars, das im eingeben Ausdruck Variablen belegt und die Zahl der belegten Variablen im dritten Argument zurückliefert. Ist diese Zahl größer als 0, schlägt durch Cut/Fail die Anfrage fehl, ansonsten kommen die übrigen Regeln zur Anwendung.

Die beiden letzten Regeln im Programm dienen nur einer einfachen Schreibweise für mehrfache Ableitungen. Denn zum Beispiel die vierte Ableitung wird durch vierfache Anwendung der Ableitung auf denselben Ausdruck erhalten. Dies kann leicht auf die angegebene Weise rekursiv formuliert werden. In der Anwendung muß die Nummer der Ableitung als drittes Argument angegeben sein.

Ganz wichtig ist die Reihenfolge, in der die Prolog-Regeln stehen. Würden wir z.B. die Kettenregel an den Anfang stellen, gäbe es bereits bei einer so einfachen Funktion wie $\sin(x)$ Probleme. Denn durch $S=[F,U|_]$ wird ja in U das Argument von sin geliefert, das ist x. Als nächstes wird (da $U=x$) wieder $\text{dif}(\sin(x),x,A)$ aufgerufen, es entstünde also eine Endlosschleife. Genauso ginge es bei unbekannten Funktionen.

Die Kettenregel ist also am besten am Ende aufgehoben. So können vorher alle Fälle ausgefiltert werden, für die die Kettenregel nicht zuständig ist. Dies geschieht durch die Regeln für die elementaren Funktionen sowie für die unbekannten Funktionen.

Zwei Regeln gibt es für Potenz-Ausdrücke. Hier steht die Regel $(x^n)' = n \cdot x^{n-1}$ am Anfang. Die Fälle, für die diese Regel nicht vorgesehen ist, werden durch „not in(X,N)“ extrahiert. Für diese kommt die Regel $(u^v)' = \exp(v \cdot \ln(u))'$ in Frage.

Die Leistungsfähigkeit und die Mängel des Programms sind an einigen Beispielen in Bild 2 und in Bild 3 gezeigt. Dort werden Dialoge mit dem Prolog-System geführt. In Bild 2 sehen wir die Nützlichkeit der Regel $f(x)' = fS(x)$. Mit ihr erhalten wir nämlich auch die allgemeinen Ableitungsregeln, zunächst also Additions- und Subtraktionsregel, dann Produkt- und Kettenregel.

```
?-dif(f(x)+g(x),x,D).
D = fS(x) + gS(x)

?-dif(f(x)-g(x),x,D).
D = fS(x) - gS(x)

?-dif(f(x)*g(x),x,D).
D = fS(x) * g(x) + f(x) * gS(x)

?-dif(f(g(x)),x,D).
D = fS(g(x)) * gS(x)

?-dif(x^n,x,D).
D = n * x^(n-1)
```

Bild 2. Berechnung der allgemeinen Ableitungsregeln

Schließlich erfahren wir noch die Ableitung von x^n .

In Bild 3 stehen einige Anwendungsfälle, und das offenbart sich gleich die Schwäche des Systems in der momentanen Version. Obwohl natürlich alle Resultate korrekt sind, können wir mit einigen nicht sofort etwas anfangen, weil sie viel zu kompliziert sind; sie müssen noch vereinfacht werden. Dabei hilft ebenfalls Prolog.

Vereinfachen symbolischer Ausdrücke

Für einen Prolog-Interpreter ist das zwar nicht ganz so einfach, aber dennoch auch machbar. Wir wollen uns daher überlegen, wie symbolische Ausdrücke in Prolog vereinfacht werden können. Sehen wir uns zunächst ein Beispiel an: Zu vereinfachen sei das Produkt $5 \cdot a \cdot 3 \cdot a^3$. Es ist klar: Man kann 5 und 3 zu 15 zusammenfassen sowie a und a^3 zu a^4 . Wie aber kann ein Computer-Programm dies entscheiden? Es müßte zum Beispiel die 5 betrachten und den gesamten Rest durchsehen, ob sich darin etwas „Passendes“ befindet. Es würde die 3 finden. Das Produkt (also 15) würde dann zum Beispiel an den Anfang

geschrieben, 5 und 3 gelöscht. Zusammenfassen von a und a^3 wäre nach dieser Methode auch kein Problem mehr. Das Resultat wäre also $15 \cdot a^4$.

Was würde nun bei $a \cdot 5 \cdot a^3 \cdot 3$ passieren? Nun, wie man sich leicht überlegen kann, käme $4 \cdot 15$ heraus. Beide sind natürlich gleich. Dennoch würde das Programm vor nahezu unüberwindliche Probleme gestellt, sollte es nun die Summe $15 \cdot a^4 + 4 \cdot 15$ zusammenfassen, was wiederum für den Menschen ein Kinderspiel ist. Hinzu käme noch, daß das angesprochene Löschen und Hinzufügen der Faktoren nicht so einfach zu realisieren ist. Ein weitaus besserer Ansatz ist es, die auftretenden Ausdrücke zunächst zu sortieren: Damit kämen wir je nach verwendeter Ordnung in beiden Fällen zum Beispiel auf $3 \cdot 5 \cdot a \cdot a^3$. Nun bräuchte das Programm nur noch benachbarte Faktoren zu betrachten. Das Resultat wäre identisch $15 \cdot a^4$, auch die Summe könnte leicht zu $30 \cdot a^4$ zusammengefaßt werden.

Wie Zahlen untereinander geordnet werden, ist klar. Auch für Text-Atome kann man sich eine Ordnung vorstellen, nämlich die alphabetische. Was geschieht aber, wenn Zahlen und Text-Atome gemischt auftreten? Beide können außerdem noch einen Exponenten besitzen.

Eine Ordnung für Strukturen

Es hilft also nichts: Wir müssen zuerst eine Ordnung zwischen allen auftretenden Strukturen erklären. Dazu definieren wir ein Prädikat $\text{lt}(X,Y)$ (less than), das uns angeben soll, ob X kleiner als Y ist (siehe Listing in Bild 4). Aufgrund dieser Eigenschaft können wir beim Sortieren entscheiden, ob zum Beispiel 3 links von a^5 stehen muß oder nicht.

```
?-dif(sin(exp(x)),x,D).
D = cos(exp(x)) * exp(x)

?-dif(cos(ln(x)),x,D).
D = (- sin(ln(x))) * (1 / x)

?-dif(x^n,x,D).
D = exp(x * ln(x)) * (1 * ln(x) + 1 / x * x)

?-dif(2^x,x,D).
D = exp(x * ln(2)) * (1 * ln(2) + 1 / 2 * 0 * x)

?-dif(x^2+x+1,x,D).
D = 2 * x^(2-1) + 1 + 0

?-dif(a ln(x),x,4,D).
D = - (- sin(x))

?-dif(x^2,x,2,D).
D = 0 * x^(2-1) + (2-1) * x^(2-1-1) = 2

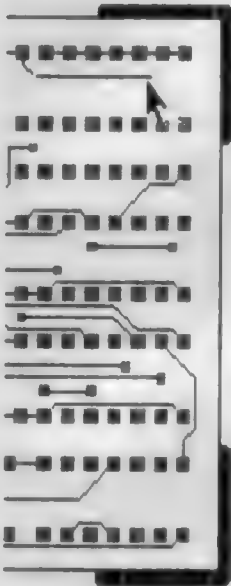
?-dif(ln(x),x,2,D).
D = 0 * x^-1 + -1 * x^(-1-1) = 1
```

Bild 3. Einige Anwendungen des Ableitungsprogramms

Mit SHAMROCK SOFTWARE können Sie zum Beispiel ...

... Leiterplatten entwickeln ...

Board: DEMO



Add
Assign
Circle
Copy
Delete
Display
Grid
Group
Layer
Move
Name
Open
Pin
Quit
Rotate
Script
Show
Snd
Value
Window
;
Wire
Write
Route

Der neue interaktive, mausgesteuerte Grafik-Editor EAGLE bietet eine komfortable Benutzerführung durch Pop-Up-Menüs. Die Auflösung beträgt 1/1000 Zoll, das Raster kann auf Zoll, mm und Mil umgeschaltet werden. Bauelemente können platziert, verschoben, gedreht oder gelöscht werden. Das Verschieben und Kopieren von ganzen Schaltungsteilen ist kein Problem. Änderungen können über eine Undo- bzw. Redo-Funktion rückgängig gemacht werden. EAGLE enthält auch einen komfortablen Handrouter, mit dem Signale manuell verdrahtet werden können, die maximale Zeichenfläche beträgt 64 x 64 Zoll (ca. 1600 x 1600 mm). Die Breite der Leiterbahnen sowie die Größe der Lötäugen sind frei wählbar. Es können bis zu 255 Layer frei definiert werden. Die Bauteilbibliotheken lassen sich auf einfachste Weise editieren bzw. erweitern, die gängigsten Bauteilformen in konventioneller und SMD-Technik werden mitgeliefert. Für eine Platine können bis zu 255 Bibliotheken eingesetzt werden, die jeweils bis zu 65 000 Makros enthalten können (einzige Einschränkung: die Größe des Arbeitsspeichers).

Die Kompatibilität zum AUTOROUTER 3 ist gewahrt: Der Grafik-Editor ermöglicht die Platzierung der Bauteile für den Autorouter 3, die Vorverlegung von Leiterbahnen, die Definition von Sperrflächen sowie die Bearbeitung von Makros. Vom AUTOROUTER 3 geroutete Layouts können nachbearbeitet werden.

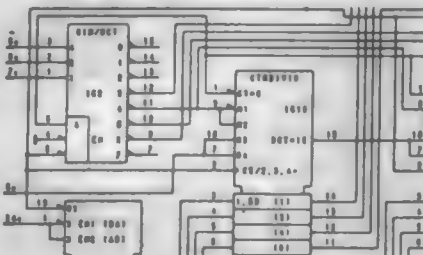
Ausgabe des Layouts: Postscript-fähige Laserdrucker, HP-GL-Plotter, Epson-kompatible Drucker, NEC-Pinwriter (24 Nadeln) und Kompatible.

EAGLE läuft auf kompatiblen PCs/ATs mit EGA- oder VGA-Grafik und Farb-Monitor sowie Microsoft-kompatibler (!) Maus **764 DM**

Mit AUTOROUTER 3 können ein- oder zweiseitige Platinen bis zur Größe einer Doppel-Europakarte mit Hilfe eines hervorragend optimierten Routing-Algorithmus automatisch im 1/20-Zoll-Raster entflochten werden. Busstrukturen werden bevorzugt behandelt, auf Löt- und Bestückungsseite werden Vorzugsrichtungen eingehalten. AUTOROUTER 3 erzeugt das Layout für beide Seiten, Bohrschablone, Bestückungsplan sowie ein Gummiband-Verbindungsschema. Eine von SHAMROCK-CAD aus dem Schaltplan erstellte Signal-Verbindungsliste kann direkt verarbeitet werden.

AUTOROUTER 3 für PCs/ATs mit CGA-, HGC-, EGA- od. VGA-Grafik **764 DM**

... und Schaltbilder zeichnen



SHAMROCK-CAD ist dafür prädestiniert: Eine große Bibliothek an Standard-Schaltzeichen (ca. 800 Symbole TTL-Bausteine, diskrete Bauelemente) wird mitgeliefert. Die Bibliotheken können nach Bedarf erstellt, erweitert oder modifiziert werden. Ganze Zeichnungsbereiche können verschoben, kopiert oder gelöscht werden. Die Zeichnungen können mit komfortablen Ausgabe-programmen auf Drucker oder Plotter ausgegeben werden. Die Benutzung einer Microsoft-kompatiblen Maus ist optional. Ein Signal-Analyse-Programm erstellt aus der Zeichnung die Signaletzliste für AUTOROUTER 3.

SHAMROCK-CAD für PCs/ATs mit CGA-, HGC- oder EGA-Grafik . . . **495 DM**



Und das ist noch keineswegs alles! Bitte fordern Sie unseren kostenlosen Gesamtkatalog an. (Alle genannten Programme sind auf Industriestandard-kompatiblen MS-DOS-Rechnern lauffähig.)

SHAMROCK SOFTWARE Vertrieb GmbH

Karlstraße 35, 8000 München 2, Telefon 0 89/59 54 68

An unterster Stelle der Objekte sollen die Zahlen stehen. Diesen ist die erste Regel gewidmet. Zuerst wird getestet, ob die beiden Argumente tatsächlich Zahlen sind. Ist das der Fall, kann die Entscheidung, ob $X < Y$ gilt, bereits getroffen werden. Deshalb muß nun ein Cut stehen. Sonst würde bei negativem Ausgang der Frage $X < Y$ die nächste Regel aufgesucht. Dies ist im übrigen bei allen Regeln für It so: Sobald der Typ der Objekte erkannt ist, kommt aus dem genannten Grund der Cut.

Jede Zahl soll kleiner als jedes Text-Atom sein, das ist die Aussage der nächsten Regel. Die Frage It(X,Y) wird also auf falsch erkannt, falls X ein Text-Atom und Y eine Zahl ist. Dieser Fall muß ebenfalls durch eine Regel erfaßt werden, da sonst die letzte Regel zur Anwendung käme. Diese würde unter Umständen ein falsches Ergebnis liefern. Nun kommen wir zur Ordnung der Text-Atome untereinander. Hierfür wünschen wir, wie bereits erwähnt, eine alphabetische Ordnung. Eine solche ist in Prolog meist nicht implementiert. Dafür gibt es aber ein Prädikat name, das ein Text-Atom in eine ASCII-Liste umwandelt. In einer solchen Liste stehen die ASCII-Werte der

im Text-Atom vorkommenden Zeichen. Wir können also beide Text-Atome in die entsprechenden ASCII-Listen wandeln und diese Element für Element vergleichen.

Den Vergleich der ASCII-Listen übernimmt das Prädikat It1. Es ist rekursiv definiert. Zunächst die Abbruchbedingungen: Die leere Liste ist kleiner als jede nicht-leere Liste. Und jede nicht-leere Liste ist nicht kleiner als die leere Liste. Inbegriffen ist hier die Aussage: Die leere Liste ist nicht kleiner als die leere Liste. Das muß auch so sein, denn dieser Fall tritt als Abbruchbedingung auf, wenn die beiden Text-Atome und damit auch die beiden ASCII-Listen identisch sind.

Wenn nun die ersten Elemente beider Listen gleich sind, ist eine Entscheidung noch nicht möglich, und die anderen Elemente müssen herangezogen werden. Sind die ersten Elemente A und B hingegen ungleich, kann die Rekursion sofort abgebrochen werden. In diesem Fall hängt das Ergebnis davon ab, ob A kleiner als B ist oder nicht. Nun kommen wir zu den Potenzen. Wie wir in dem einführenden Beispiel gesehen haben, spielt hier nur die Basis eine Rolle. Ist die Basis zweier Potenzen identisch,

können sie zusammengefaßt werden. Deswegen führen wir das Problem der Potenzen jeweils auf die Basis zurück. Drei Fälle haben wir hier abzudecken: Entweder nur das erste Argument ist eine Potenz oder nur das zweite oder beide.

Im folgenden sollen alle bisher betrachteten Strukturen (Zahlen, Text-Atome und Potenzen) zusammenfassend „Monome“ genannt werden. Hilfreich ist hierfür das vordefinierte Prädikat atomic, welches wahr ist, falls das Argument eine Zahl oder ein Text-Atom ist. Eine Struktur ist also ein Monom, wenn sie entweder vom Typ atomic ist oder die Form Y^N mit beliebigen Y und N hat.

Ausdrücke und Terme

Fahren wir mit den It-Regeln fort. Wir legen nun fest, daß atomic-Ausdrücke immer kleiner als alle übrigen Strukturen sein sollen. Hierzu müssen wir aber wissen, was an sonstigen Strukturen überhaupt noch möglich ist. Es sind dies „Ausdrücke“ und „Terme“. „Ausdrücke“ sind hierbei Summen, wobei auch einelementige Summen erlaubt sind. Unter einem „Term“ verstehen wir ein Produkt, ein einelementiges Produkt ist ebenfalls zugelassen.

Soll nun der Ausdruck $a+b$ größer als der Term $x*y$ sein oder nicht? Es ist recht schwierig, sich auch für Terme und Ausdrücke eine Ordnung zu überlegen. Deshalb wandeln wir Terme und Ausdrücke nach bestimmten Kriterien in jeweils eine Liste von Zahlen um. Diese können dann mit dem bereits definierten Prädikat It1 verglichen werden. Für die Umwandlung in die Liste ist das Prädikat str_list zuständig. Wie entsteht diese Liste? Zahlen und Text-Atome können mittels name direkt in eine ASCII-Liste verwandelt werden. In anderen Fällen muß die Struktur genauer untersucht werden. Hier hilft der Operator $=..$, der Operatoren und Operanden trennt. Er liefert eine Liste, deren erstes Element der Operator mit der niedrigsten Bindungszahl (die äußere Funktion) ist. Die weiteren Elemente werden von den Operanden bzw. dem Argument der Funktion eingenommen.

Wenn also $E=..[P,A]$ wahr ist, dann muß es sich bei E um einen Funktionsaufruf handeln; P ist der Name der Funktion und A das Argument. In diesem Fall werden beide weiter zerlegt und die Resultate durch append aneinandergehängt. Wäre $E=\sin(x)$, würde $P=\sin$ und $A=x$ erkannt. P würde direkt mit name zerlegt und A im rekursiven Aufruf von der ersten Regel. Das Gesamtergebnis wäre [115,105,110,120].

Liegt kein Funktionsaufruf vor, muß es sich um einen Operator wie + oder * handeln.

```

It(X,Y):-number(X),number(Y),!,X<Y.           % Zahlen
It(X,Y):-number(X),atom(Y),!.                 % Zahl < Text-Atom
It(X,Y):-atom(X),number(Y),!,fail.             % Text-Atom > Zahl
It(X,Y):-atom(X),atom(Y),!,name(X,L),          % Atome
        name(Y,M),It1(L,M).
It(X,Y^N):-!,It(X,Y).                          % Potenzen
It(X^M,Y):-!,It(X,Y).
It(X^M,Y^N):-!,It(X,Y).
It(X,Y):-atomic(X),!.                          % Atomic < Sonstiges
It(X,Y):-atomic(Y),!,fail.                     % Sonstiges > Atomic
It(X,Y):-str_list(X,L),str_list(Y,M),!,It1(L,M). % Sonstiges
It([],M):-!,not M= [].                         % ASCII-Listen
It(L,[],_):-!,fail.
It([A:_,_],[_:_,_]):-!,It1(L,M).
It([A:_,_],[_:_,_]):-A<B.
str_list(C,L):-atomic(C),name(C,L).            % Struktur --> Liste
str_list(E,L):-E=..[P,A],name(P,L1),
        expr_list(A,L2),append(L1,L2,L).
str_list(E,L):-E=..[P,A,B],name(P,L1),
        expr_list(A,M),expr_list(B,N),
        append(L1,M,L1),append(L1,N,L).

```

Bild 4. Das Prädikat It definiert eine Ordnung

```

norm_expr(R-X,S+(-X)):-!,norm_expr(R,S).      % Ausdruck normalisieren
norm_expr(R+X,S+X):-!,norm_expr(R,S).
norm_expr(X,X).
norm_term(U,T):-norm_t1(U,V),norm_t2(V,T).    % Term normalisieren
norm_t1(R/X,S*X^(-1)):-!,norm_t1(R,S).        % Division beseitigen
norm_t1(R*X,S*X):-!,norm_t1(R,S).
norm_t1(X,X).
norm_t2(R*(-X),S*(-1)*X):-!,norm_t2(R,S).     % Minus abtrennen
norm_t2(R*X,S*X):-!,norm_t2(R,S).
norm_t2(-X,-1*X).
norm_t2(X,X).
norm_mono((U^X)^Y,V):-!,norm_mono(U^(Y*X),V). % Potenz
norm_mono(X,X).                               % normalisieren

```

Bild 5. Regeln zur Normalisierung

Dann ist $E = \dots[P, A, B]$ wahr. Die Operanden A und B werden rekursiv zerlegt, der Operator mit name. Die Resultate werden wie eben mit append zusammengefügt. Zum Beispiel ergibt bei $E = a * b + c * d$ die Unifikation $P = '+'$, $A = a * b$ und $B = c * d$. P wird direkt in die Liste [43] umgewandelt, 43 ist der ASCII-Code von +. Für $a * b$ wird str_list erneut aufgerufen. Hier wird der Operator * erkannt, dessen ASCII-Code ist 42. Für die Argumente a und b ergibt die erste Regel [97] und [98], so daß für $a * b$ die Liste [42, 97, 98] herauskommt. Analog ergibt $c * d$ die Liste [42, 99, 100]. Zusammen mit der 43 erhalten wir also [43, 42, 97, 98, 42, 99, 100].

Terme und Ausdrücke werden also zunächst in Listen umgewandelt, um dann mit lt1 verglichen zu werden. Damit ist die Ordnung vollständig, aber es steht noch das Sortieren aus. Es gibt zwei verschiedene Operatoren, die einen Term bilden können, nämlich * und /. Wollten wir aus einem Term zwei Faktoren isolieren, um sie vergleichen zu können, müßten wir eine vierfache Fallunterscheidung treffen. Es kämen nämlich $R * X * Y$, $R * X / Y$, $R / X * Y$ und $R / X / Y$ in Frage. Hierbei sind X und Y die zu vergleichenden Faktoren und R der Restterm.

Normalisieren

Dasselbe Problem ergibt sich bei Ausdrücken mit + und -. Was ist hier zu tun? Wir stellen bei Ausdrücken und Termen einfach eine „Normalform“ her, in der / bzw. - verschwunden sind. Diese Strukturen dürfen natürlich nur in äquivalente Strukturen verwandelt werden, was durch Hinzufügen des Exponenten -1 bzw. des Vorzeichens - erreicht wird. Der Term $a/b * c/d$ lautet normalisiert also $a * b^{-1} * c * d^{-1}$, der Ausdruck $a - b - c$ wird zu $a + (-b) + (-c)$.

norm_expr im Listing von Bild 5 normalisiert Ausdrücke. Wir haben nur Zugriff auf das rechts stehende Element eines Ausdrucks, und zwar durch Unifikation mit $R + X$ bzw. $R - X$. Das Prädikat muß daher rekursiv aufgebaut sein. Im Fall $R - X$ muß der Rest R untersucht werden, das Resultat sei S. Dann ist das Gesamtergebnis $S + (-X)$. Im Fall $R + X$ ist eine Umwandlung nicht notwendig. Die letzte Regel im Listing dient als Abbruchbedingung.

Das Normalisieren eines Terms umfaßt nicht nur das Beseitigen der Division. Dies funktioniert analog zur Normalisierung von Ausdrücken und wird von norm_t1 durchgeführt. Zusätzlich behandeln wir hier den Fall negativer Faktoren. $b * a * (-a)$ wird demnach zu $b * a * (-1) * a$. Warum ist das

Bild 6. Der Programmteil zum Sortieren von Ausdrücken und Termen

```
sort_expr(L,N):-cha_expr(L,M),!,sort_expr(M,N). % Ausdruck sortieren
sort_expr(L,L).

cha_expr(R+X+Y,R+Y+X):-lt(X,Y),!. % Austauschen
cha_expr(X+Y,Y+X):-not X=_+,!,lt(X,Y).
cha_expr(R+X,S+X):-!,cha_expr(R,S).

sort_term(L,N):-cha_term(L,M),!,sort_term(M,N). % Term sortieren
sort_term(L,L).

cha_term(R*X*Y,R*Y*X):-lt(X,Y),!. % Austauschen
cha_term(X*Y,Y*X):-not X=_*,!,lt(X,Y).
cha_term(R*X,S*X):-!,cha_term(R,S).
```

erforderlich? Nun, das hängt mit der definierten Ordnung von Termen und Ausdrücken zusammen. $(-b)$ kann nicht als Monom erkannt werden, auch wenn b ein Monom sein sollte. Das liegt an dem Vorzeichen. Sortieren von $b * a * (-a)$ in absteigender Folge würde also $(-a) * b * a$ ergeben. $(-a)$ und a könnten nicht zusammengefaßt werden, weil sie nicht nebeneinander stehen. Jedoch $b * a * (-1) * a$ wird zu $b * a * a * (-1)$ sortiert, womit wir dieses Problem beseitigt haben. Das Abtrennen der Vorzeichen übernimmt das Prädikat norm_t2. norm_term ruft nacheinander norm_t1 und norm_t2 auf.

Schließlich müssen auch Potenzen normalisiert werden. Das wird zum Beispiel notwendig, wenn Potenzen wie $(a^n)^m$ auftreten. Wenn der Term $a * b / a^3$ normalisiert wird, ist $a * b * (a^3)^{-1}$ das Resultat. Für die Ordnung ergeben sich daraus keine Probleme, $a * b * (a^3)^{-1}$ wird zu $b * a * (a^3)^{-1}$ sortiert, da die Ordnung einer Potenz gleich der Ordnung der Basis ist. Das Zusammenfassen von a und $(a^3)^{-1}$ wäre jedoch nicht ganz einfach. Das Normalisieren von Potenzen umfaßt also das Zusammenfassen von mehreren Exponenten zu einem, denn $(a^n)^m$ ist gleich $a^{(m*n)}$. Das entsprechende Prädikat norm_mono wird rekursiv formuliert, da dies auch in höheren Stufen erlaubt ist. $((x^2)^3)^4$ wird demnach zu $x^{(4*3*2)}$. Es ist nun nicht notwendig, einen möglichst raffinierten Sortier-Algorithmus wie Quicksort oder Heapsort zu implementieren. Da die Terme selten mehr als 10 Faktoren haben werden, können wir es uns wiederum einfach machen. Wir müssen auch bedenken, daß wir aufgrund der unterschiedlichen Zugriffsformen den Algorithmus zweimal implementieren, und zwar für Ausdrücke und Terme. So ist es ganz praktisch, daß er nur fünf Regeln umfaßt (siehe Listing in Bild 6).

Der Zugriff erfolgt in beiden Fällen von rechts. Zum Beispiel ergibt die Unifikation von $a * b * c * d$ mit $R * X * Y$ das Resultat $R = a * b$, $X = c$, $Y = d$. X und Y beinhalten also in jedem Fall die Faktoren, die verglichen werden können.

Der Sortier-Algorithmus

Wir besprechen den Sortier-Algorithmus nun für Ausdrücke. Dabei wird jeweils in absteigender Ordnung sortiert, die Zahlen als kleinste Elemente stehen schließlich also ganz rechts. Wir könnten dies auch genau umgekehrt tun, diese Ordnung wird sich jedoch später als sinnvoll erweisen. Zunächst benötigen wir ein Prädikat cha_expr, das im Ausdruck zwei nebeneinanderstehende Terme sucht, die vertauscht werden können. Dann wird der modifizierte Ausdruck im zweiten Argument zurückgeliefert. Gibt es keine zwei Terme, die vertauscht werden können, wird das Prädikat falsch.

X und Y sind in der ersten Regel die ganz rechts stehenden Terme. Ist X kleiner als Y, müssen beide vertauscht werden. Analog funktioniert das, falls der gesamte Ausdruck nur aus zwei Termen besteht, wie das in der zweiten Regel der Fall ist. Wenn beide Regeln nicht anwendbar sind, wird der rechts stehende Term abgetrennt und das Prädikat rekursiv auf den Rest angewandt. sort_expr versucht mittels cha_expr zwei Terme zu vertauschen. Gelingt dies, wird sort_expr erneut für das Resultat aufgerufen. Dies geht so lange, bis irgendwann cha_expr falsch ist. Dann sind keine Terme mehr vertauschbar, und die Ausführung endet hier. Bei fehlendem Cut in dieser Regel würden neben der korrekten Lösung jede

```
?-lt(a*b,a+b).
yes

?-lt(x^2-y,x^2).
no

?-norm_term(a/b*c^3*(-d),S).
S = a * b ^ -1 * c ^ 3 * -1 * d

?-norm_mono((a^2)^3,S).
S = a ^ (3 * 2)

?-sort_term(a*b*a^3*(a^3)^2*b,S).
S = b * b * a * a ^ 3 * (a ^ 3) ^ 2

?-sort_expr(x^4+2*x^3+b*x^2+4,S).
S = b * x ^ 2 + 2 * x ^ 3 + x ^ 4 + 4

?-norm_term(a/b/a*b,L),sort_term(L,M).
L = a * b ^ -1 * a * a ^ -1 * b
M = b ^ -1 * b * a * a ^ -1
```

Bild 7. Die neuen Prädikate müssen zeigen, was sie können

Menge falsche Regeln produziert. Der Algorithmus für Terme wird nun ganz einfach dadurch erhalten, daß jedes + durch * ersetzt wird.

Mit der Definition einer Ordnungsrelation sowie den Prädikaten zum Normalisieren und Sortieren der auftretenden Strukturen haben wir nun die Voraussetzungen geschaffen, um die wichtigsten Vereinfachungen in arithmetischen Ausdrücken vornehmen zu können. Im Listing von *Bild 7* sind in Dialogform noch einige Beispiele für die definierten Prädikate angegeben.

Zusammenfassen sortierter Elemente

Für Ausdrücke und Terme benötigen wir noch einige Hilfs-Prädikate, die im Listing von *Bild 8* zu finden sind. Da geht es zunächst um das Verbinden von Ausdrücken. Hierzu muß ein eigenes Prädikat geschrieben werden, weil wir bei Ausdrücken nur Zugriff auf den rechts stehenden Term haben. Wir müssen daher den zweiten angegebenen Ausdruck rekursiv durchgehen, bis der erste vorne angehängt werden kann. Analog funktioniert natürlich das Verbinden von Termen.

Das Prädikat *termexpr* wandelt einen Term in einen Ausdruck um, indem es alle Multiplikationssymbole durch + ersetzt. Auch dies geschieht rekursiv, das rechts stehende Element wird jeweils abgespalten. Die Abbruchbedingung ist erfüllt, wenn nur noch ein Faktor übrig ist. In diesem Fall ist natürlich das Resultat gleich der Eingabe. Nun zum Vereinfachen: Der entsprechende Programmteil ist im Listing von *Bild 9* angegeben. Wir benötigen die drei Prädikate *red_expr*, *red_term* und *red_mono*, welche Ausdrücke, Terme und Monome vereinfachen (*red* steht für reduzieren). Alle haben zwei Argumente, das erste ist die zu vereinfachende Struktur, das zweite das Resultat. Der Aufbau aller drei Prädikate ist identisch, er soll am Beispiel von *red_expr* besprochen werden. Zunächst wird der Ausdruck normalisiert. Als nächstes werden die Einzelteile behandelt, das sind die Terme, die den Ausdruck bilden. Hierfür ist *red_e1* zuständig. Der Ausdruck wird rekursiv Term für Term durchgegangen und für jeden Term *red_term* aufgerufen. Die so vereinfachten Terme werden nun sortiert, schließlich werden einige für Ausdrücke vorgesehene Methoden angewandt, was *red_e2* erledigt. *red_term* kann man analog betrachten. Für *red_mono* entfällt das Sortieren, da die Potenzbildung nicht vertauschbar ist. Alle drei Prädikate sind rekursiv verknüpft. Das ist unbedingt erforderlich, denn eine Potenz kann zum Beispiel

einen Ausdruck als Basis haben. Auch dann soll die Basis ja vereinfacht werden. *red_mono* ruft über den Umweg *red_m1* wieder *red_expr* auf.

Wie die drei Begriffe Ausdruck, Term und Monom zusammenhängen, gibt folgende

Grammatik wieder:

Ausdruck: = Ausdruck+Term|Term

Term: = Term*Faktor|Faktor

Faktor: = -Ausdruck|Ausdruck^Ausdruck|
Ausdruck|Funktion(Ausdruck)|
Textatom|Zahl

```
app_expr(L,X,L+X):-not X=_,!,
app_expr(L,M+X,N+X):-app_expr(L,M,N).

app_term(L,X,L*X):-not X=_,!,
app_term(L,M*X,N*X):-app_term(L,M,N).

termexpr(X,Y):-not X=_,!,Y=X.
termexpr(R*X,S+X):-termexpr(R,S).
```

Bild 8. Einige Hilfsprädikate

```
:-op(700,xfx,':=').
R:=U :- red(U,R).

red(U,V):-numbervars(U,0,Z),Z>0,! ,fail.
red(U,V):-red_expr(U,V).

red_expr(U,V):-norm_expr(U,N),red_e1(N,M),eort_expr(M,S),red_e2(S,V),!.

red_e1(R+X,T):-!,red_e1(R,S),red_term(X,Y),app_expr(S,Y,T).
red_e1(X,Y):-red_term(X,Y).

red_e2(R+X+Y,S):-red_sum(X+Y,Z),!,red_e2(R+Z,S).
red_e2(X+Y,Z):-red_sum(X+Y,Z),!.
red_e2(R+X,S+X):-!,red_e2(R,S).
red_e2(X,X).

red_sum(X+Y,Z):-number(X),number(Y),Z is X+Y,!.
red_sum(X+X,Z):-!,red_term(X*2,Z).
red_sum(X+0,X):-!.
red_sum(U+X+Y,V):-number(X),number(Y),!,Z is X+Y,(Z=0,V=0;V=U+Z).
red_sum(U+U+Y,V):-number(Y),!,Z is Y+1,(Z=0,V=0;true,V=U+Z).
red_sum(U+X+U,V):-number(X),!,Z is X+1,(Z=0,V=0;true,V=U+Z).

red_term(U,V):-norm_term(U,N),red_t1(N,M),eort_term(M,S),red_t2(S,V),!.

red_t1(R*X,T):-!,red_t1(R,S),red_mono(X,Y),app_term(S,Y,T).
red_t1(X,Y):-red_mono(X,Y).

red_t2(R*X*Y,S):-red_prod(X*Y,Z),!,red_t2(R*Z,S).
red_t2(X*Y,Z):-red_prod(X*Y,Z),!.
red_t2(R*X,S*X):-!,red_t2(R,S).
red_t2(X,X).

red_prod(X*Y,Z):-number(X),number(Y),Z is X*Y,!.
red_prod(X^N*X^M,S):-!,red_expr(X^(N+M),S).
red_prod(X^N*X,S):-!,red_expr(X^(N+1),S).
red_prod(X*X^M,S):-!,red_expr(X^(M+1),S).
red_prod(X*X,X^2):-!.
red_prod(X*0,0):-!.
red_prod(X*1,X):-!.

red_mono(U,V):-norm_mono(U,N),red_m1(N,M),red_m2(M,V),!.

red_m1(-U,V):-!,red_term(-1*U,V).
red_m1(X^N,Y^M):-!,red_expr(X^N),red_expr(Y^M).
red_m1(U,V):-U=..[F,X,Y],!,red_expr(U,V).
red_m1(U,V):-U=..[F,X],!,red_expr(X,Z),V=..[F,Z].
red_m1(X,X).

red_m2(X^N,S):-pow(X,N,S),!.
red_m2(sin(0),0):-!.
red_m2(cos(0),1):-!.
red_m2(exp(0),1):-!.
red_m2(ln(1),0):-!.
red_m2(exp(ln(U)),U):-!.
red_m2(exp(U),S):-not U=_,red_ln(U,L,T),!,R=L^T,red_mono(R,S).
red_m2(ln(exp(X)),X):-!.
red_m2(X,X).

red_ln(T*ln(L),L,S):-!,termexpr(T,S).
red_ln(ln(L),L,0):-!.
red_ln(R*X,L,S+X):-red_ln(R,L,S).

pow(X,N,S):-N<0,! ,fail.
pow(X,1,X).
pow(X,0,1).
pow(0,N,0).
pow(X,N,S):-M is N-1,pow(X,M,T),S is X*T.
```

Bild 9. Nach umfangreichen Vorbereitungen endlich die Vereinfachungen

Durch senkrechte Striche getrennte Erzeugnisse geben jeweils eigene Regeln an. Alle drei Begriffe sind natürlich nicht-terminale Symbole, d.h. zur Erhaltung eines gültigen Ausdrucks müssen sie nach den angegebenen Regeln ersetzt werden. Startsymbol ist „Ausdruck“.

Die Grammatik ist so gebaut, daß man in jedem Schritt einer Herleitung eindeutig entscheiden kann, welche Regel man anwenden muß. Steht beispielsweise in der Kette das Symbol „Ausdruck“, so müssen wir untersuchen, ob an dieser Stelle ein + erzeugt werden muß oder nicht. Im ersten Fall müssen wir „Ausdruck“ durch „Ausdruck+Term“ ersetzen, ansonsten durch „Term“. Genauso ist das bei den anderen Regeln.

Die Verknüpfung ist durch die Prädikate `red_e1`, `red_t1` und `red_m1` realisiert. Sehen wir uns zunächst `red_e1` an. Zwei Regeln gibt es zur Ersetzung von „Ausdruck“ in der Grammatik. Auch für `red_e1` sind zwei Regeln vorhanden, die den Grammatik-Regeln entsprechen. Die erste ist nur anwendbar, wenn der Ausdruck die Form $R+X$ hat. In diesem Fall wird das Prädikat für R wieder aufgerufen, da sich darin weitere Pluszeichen befinden können. Bei X ist das nicht der Fall, hier kann gemäß unserer Grammatik direkt `red_term` aufgerufen werden. Die beiden Resultate sind Ausdrücke, sie werden durch das bereits angesprochene `app_expr` verbunden.

Ist die erste Regel nicht anwendbar, muß es sich bei dem Ausdruck um einen Term handeln, das Argument kann direkt an `red_term` übergeben werden. Wie `red_term` arbeitet, haben wir ja bereits besprochen. Nach dem Normalisieren des Terms ruft es `red_t1` auf. Dieses ist analog zu `red_e1` aufgebaut, auch die Grammatik-Regeln für „Term“ sind ja genau analog zu denen für „Ausdruck“.

Interessanter ist `red_m1`; es wird über `red_mono` von `red_t1` aufgerufen. Hier gibt es mehrere Möglichkeiten. U kann ein negatives Vorzeichen haben. Normalerweise wird es durch das Normalisieren des Terms beseitigt. Es kann aber sein, daß der Ausdruck in Klammern stand, so daß das Vorzeichen nicht erkannt wurde. Dies entspricht der ersten Regel unserer Grammatik für „Faktor“.

Für eine Potenz werden Basis und Exponent getrennt behandelt, das gibt die darauf folgende Regel an. Die nächste Grammatik-Regel sagt aus: „Faktor:=Ausdruck“. Zur Anwendung muß die Struktur zunächst als Ausdruck erkannt werden. Dies geschieht mit `-..`. Die resultierende Liste muß drei Argumente haben, die Art des Operators ist unerheblich, es kann sich aber nur um +,

-, * oder / handeln, ^ kam bereits in der vorherigen Regel an die Reihe. Es ist auch egal, welche Form die Operanden haben, der gesamte Ausdruck kann an `red_expr` übergeben werden.

Bei Funktionen wird das Argument als Ausdruck vereinfacht, der Funktionsname wird natürlich unverändert gelassen. Wenn alle diese Fälle nicht eingetreten sind, muß es sich um eine Zahl oder ein Text-Atom handeln. Auch an diesen wird nichts geändert, daher sind bei der letzten Regel beide Argumente gleich.

Spezielle Vereinfachungsregeln

`red_e2`, `red_t2` und `red_m2` behandeln spezielle Vereinfachungsregeln. Die Strukturen, die sie zu bearbeiten haben, sind bereits sortiert. Deshalb müssen sie die Strukturen nur rekursiv durchgehen und benachbarte Elemente untersuchen.

`red_e2` verwendet als Hilfsprädikat `red_sum`, das für zwei Elemente testet, ob sie zusammengefaßt werden können. Können also in $R+X+Y$ die Elemente X und Y zu Z zusammengefaßt werden, wird `red_e2` rekursiv für $R+Z$ aufgerufen. Ist ein Zusammenfassen nicht möglich, wird in der dritten Regel das rechts stehende Element abgespalten. Es gibt zwei Abbruchbedingungen: Die zweite Regel für den Fall, daß zwei übriggebliebene Elemente zusammengefaßt werden können sowie die letzte Regel für den anderen Fall.

Wie arbeitet `red_sum`? Es hat zwei Argumente; das erste ist eine Summe zweier Terme, das zweite das Resultat, der äquivalente vereinfachte Ausdruck. Sind X und Y Zahlen, können sie direkt addiert werden. Sind die beiden Terme identisch gleich X , kann man stattdessen $X*2$ schreiben. Damit haben wir auf jeden Fall einen Term hergestellt, deshalb übergeben wir dies an `red_term`. Eine 0 kann einfach weggelassen werden, das ist die Aussage der dritten Regel.

Die letzten drei Regeln gelten dem Fall, daß zwei Terme vorliegen, die bis auf einen numerischen Faktor identisch sind. Bei den beiden letzten Regeln ist einer dieser Faktoren nicht explizit vorhanden, daher gleich 1. In Z werden die Faktoren addiert. Nun kann noch der Fall eintreten, daß Z gleich 0 ist. Dann wird das gesamte Resultat auf 0 gesetzt. Ansonsten ist $U*Z$ das Ergebnis. Hier ist auch der Grund dafür, daß wir die Zahlen ans Ende jedes Terms hinsortiert haben. Da wir nur Zugriff auf dieses Ende haben, müßten wir im anderen Fall die Zahlen rekursiv aus dem Term herausholen.

`red_t2` ist absolut identisch zu `red_e2` auf-

```
?-dif(sin(exp(x)),x,D).
D = exp(x) * cos(exp(x))

?-dif(cos(ln(x)),x,D).
D = sin(ln(x)) * x^-1 = -1

?-dif(x^x,x,D).
D = (ln(x) + 1) * x^x

?-dif(2^x,x,D).
D = ln(2) * 2^x

?-dif(x^2+x+1,x,D).
D = x * 2 + 1

?-dif(sin(x),x,4,D).
D = sin(x)

?-dif(x^2,x,2,D).
D = 2

?-dif(ln(x),x,2,D).
D = x^-2 = -1
```

Bild 10. Das erweiterte System arbeitet deutlich besser

gebaut. Das für die speziellen Vereinfachungen vorgesehene Prädikat heißt hier `red_prod`. Die erste Regel von `red_prod` versucht, wie bei `red_sum`, zwei Zahlen zusammenzufassen. Bei den dann folgenden drei Regeln sind Potenzen im Spiel. Hier wird die Gleichung $x^n * x^m = x^{(n+m)}$ angewandt. Drei Regeln sind nötig, da das X auch ohne Exponent erscheinen kann, was als Exponent 1 zu werten ist. Die neu entstandene Potenz wird an `red_expr` übergeben, damit die nun erforderlichen Vereinfachungen im Exponenten ausgeführt werden. Leicht zu verstehen sind die letzten drei Regeln.

`red_m2` führt die Umformungen für Monome selbst aus. Zunächst werden Potenzen durch das Prädikat „pow“ vereinfacht. Dann folgt die Ersetzung einiger Spezialwerte der Basis-Funktionen. Interessant wird es bei den Regeln 5 und 6. Zur Durchführung der Ableitung hatten wir U^V ja in $\exp(V \cdot \ln(U))$ umgeformt. Besser lesbar ist

```
? DIF(SIN(0E^X),X);
0: 0E^X * COS(0E^X)

? DIF(COS(LN(X)),X);
0: -SIN(LN(X)) / X

? DIF(X^X,X);
0: X^X * LN(X) + X^X

? DIF(2^X,X);
0: 2^X * LN(2)

? DIF(X^2+X+1,X);
0: 1 + 2*X

? DIF(DIF(DIF(DIF(SIN(X),X),X),X),X);
0: SIN(X)

? DIF(DIF(X^2,X),X);
0: 2

? DIF(DIF(LN(X),X),X);
0: -1 / X^2
```

Bild 11. Auch ein Vergleich zum professionellen System muMATH fällt günstig aus

aber U^V , und so ist es angebracht, eine Rückumformung vorzunehmen.

Wir müssen natürlich von dem allgemeinen Fall ausgehen, bei dem das Argument von \exp ein beliebig langer Term mit einem Logarithmus-Faktor ist. Es gilt zunächst, diesen Logarithmus zu finden. Das Argument von \ln wird dann die neue Basis, der Rest des Terms der Exponent.

Das Argument von \exp darf also keine Summe sein. Das Prädikat `red_ln` führt die eben angesprochene Suche nach \ln durch. In L wird das Argument von \ln zurückgeliefert, in T der Rest. Das Resultat ist dann natürlich L^T . Dies ist eine einfache Potenz und kann wiederum durch `red_mono` vereinfacht werden.

`red_ln` kann sofort das Ergebnis liefern, wenn das Argument die Form $T \cdot \ln(L)$ hat. Der Restterm muß mit `termexpr` (siehe Bild 8) in einen Ausdruck verwandelt werden. Einfach ist es auch, wenn $\ln(L)$ das gesamte Argument von \exp ist, denn $\exp(\ln(L))$ ist gleich L . Steckt der Logarithmus zwischen anderen Faktoren, müssen wir ihn auf rekursive Art herausholen. Dies geschieht in der dritten Regel. Die Elemente, die wir rechts abspalten, werden beim Auflösen der Rekursion schon als Summe angehängt.

Auswertung von Potenzen

Bleibt nur noch das Prädikat „`pow`“ zu erklären, mit dem Potenzen ausgewertet werden. Die entsprechenden Regeln sind am Ende des Listings von Bild 9 angegeben. Im allgemeinen ist eine Auswertung nur möglich, wenn Basis und Exponent Zahlen sind. Weil viele Prolog-Versionen nur über ganze Zahlen verfügen, schließen wir negative Exponenten aus. Dies geschieht in der ersten Regel. Dann folgen einige Spezialfälle, die auch bei nicht-numerischer Basis Anwendung finden. Es gilt nämlich immer $x^{-1} = x^{-1}$, $x^0 = 1$ und $0^n = 0$. Der Spezialfall 0^0 , der nicht eindeutig definiert werden kann, ergibt hier 1, da die Regel für x^0 vor der für 0^n steht.

Hat die Potenz einen numerischen Exponenten größer als 1 sowie eine numerische Basis, wird der Wert durch die letzte Regel rekursiv berechnet. Als Abbruchbedingung dient hier die Regel $x^{-1} = x^{-1}$. Durch die Rekursion wird der Exponent jeweils um 1 verringert, bis der Exponent 1 erreicht ist. Bei gebrochenen Exponenten wird der Wert 1 nie erreicht. Durch die erste Regel wird jedoch ein Abbruch erzwingen, sobald der Exponent unter 0 gesunken ist.

Das gesamte Vereinfachungsprogramm wird durch das Prädikat `red` aufgerufen, das durch `red_expr` erklärt ist. Zuvor wird durch die erste Regel für `red` jedoch sicher-

```
?-dif(f(x)*g(x),x,2,D).
D = g(x) * f88(x) + g8(x) * f8(x) * 2 + g88(x) * f(x)

?-dif(f(x)^g(x),x,D).
D = f(x) ^ g(x) * (ln(f(x)) * g8(x) + g(x) * f(x) ^ -1 * f8(x))

?-dif(f(x)^f(x),x,D).
D = f(x) ^ f(x) * (f8(x) + ln(f(x)) * f8(x))

?-dif(1/f(x),x,D).
D = f(x) ^ -2 * f8(x) ^ -1

?-dif(ln(f(x)),x,D).
D = f(x) ^ -1 * f8(x)
```

Bild 12. Das System kann allgemeine Ableitungsregeln entdecken

gestellt, daß sich in dem Ausdruck keine Variablen befinden. Der Aufruf `red(U,R)` ist jedoch nicht besonders schön, deshalb wollen wir schließlich einen Operator `:=` definieren, so daß eine Vereinfachung durch `R:=U` durchgeführt werden kann.

Die Bindungszahl von `:=` sollte höher als die der vorkommenden Operatoren sein, wir wählen einmal 700. Er soll nicht assoziativ (`xfx`) sein, um falsche Anwendung zu vermeiden. Die Anweisung muß also lauten: `op(700,xfx,':')`. Die hierauf folgende Definition `R:=U :- red(U,R)` ruft das gewünschte Verhalten hervor.

Welchen Vorteil hat eine derartige Definition? Um den Vereinfachungsoperator für die Ableitungsregeln einsetzen zu können, müssen wir im Listing von Bild 1 nur `alle =` durch `:=` ersetzen. Damit werden ohne weiteres Zutun alle Vereinfachungen erledigt.

Probe aufs Exempel

Was fängt das erweiterte System jetzt mit den bereits im Bild 3 überprüften Ausdrücken an? Wir leiten deshalb dieselben Ausdrücke erneut ab. Das Resultat ist hier in Bild 10 angegeben. Mit den meisten Ergebnissen kann man durchaus zufrieden sein. Kleine Unschönheiten ergeben sich, wenn die Ableitung negativ ist, wie im zweiten und letzten Fall. Günstiger wäre es, wenn das Vorzeichen am Anfang des Terms stünde. Anlaß zur Kritik gibt auch der Faktor

x^{-1} , ein Divisionsstrich würde besser aussehen.

Das sind zwei Schwächen, die man noch beseitigen könnte. Derartige Rückwandlungen wären am besten im Prädikat „`red`“ nach Aufruf von „`red_expr`“ aufgehoben, da dann sämtliche Vereinfachungen bereits abgeschlossen sind.

Das Listing aus Bild 11 zeigt, wie ein Vergleich mit dem professionellen System `muMATH` ausfällt, das in der Lisp-ähnlichen Sprache `muSIMP` geschrieben ist. Es unterscheidet sich außer einigen Kleinigkeiten nur in den eben angesprochenen Punkten. Höhere Ableitungen können in `muMATH` nicht direkt sondern nur durch Verschachtelungen von `DIF` eingegeben werden.

Zum Schluß soll noch einmal auf eine ganz spezielle Fähigkeit des Systems hingewiesen werden: man kann allgemeine Ableitungsregeln ausgeben lassen. Hierzu wurde ja die Regel $f'(x) = f8(x)$ eingebaut, die bei unbekannten Funktionen angewandt wird. Bild 12 bringt einige Beispiele.

Als Fazit kann wohl gesagt werden, daß in Prolog mit einem nicht allzu hohen Aufwand Resultate erzielbar sind, die denen professioneller Systeme durchaus nahekommen. Prolog erweist sich als Sprache, die für den Umgang mit symbolischen Ausdrücken gut geeignet ist. Genauso wie hier bei der Differentiation kommt man auch auf anderen Gebieten wie Integration oder Lösen von Gleichungen mit ähnlichen Methoden relativ schnell zum Ziel. □

Spruch des Monats

„Wir dürfen unsere Kunden
nicht mit alten Produkten
zu Tode pflegen.“

Josef Dahlmann, Inhaber der DDV GmbH,
auf einer Hausmesse

Roman Gerike

Sortiertes Inhaltsverzeichnis mit Turbo 4.0

Turbo-Pascal 4.0 besitzt einen wesentlich größeren Sprachumfang als alle Vorgängerversionen. Der Compiler ist mit erweiterten Direktiven angereichert und die Technik der bedingten Übersetzung hält endlich auch in Turbo-Pascal Einzug. Das in *Bild 1* vorgestellte Programm CAT demonstriert

die Mächtigkeit der Unit DOS und die Flexibilität eines Programms durch bedingte Übersetzung.

Das Programm ist als externe Alternative zum internen MS-DOS Befehl DIR gedacht. Es kann im Suchpfad abgelegt werden, der durch den Path-Befehl (z. B. Path=C:\DOS) definiert ist. Benötigt der Anwender das Inhaltsverzeichnis eines Laufwerks, gibt er statt DIR nun CAT ein und erhält ein Inhaltsverzeichnis, in der in *Bild 2* gezeigten Form.

Dieses Verzeichnis unterscheidet sich vom Inhaltsverzeichnis des DIR-Befehls in folgenden Punkten. Das von CAT ausgegebene Verzeichnis enthält den Namen, die Größe in Bytes, die Dateiattribute und das Datum der Directory-Einträge. Im Gegensatz zum DIR-Befehl fehlt die Uhrzeit. Das Verzeichnis wird zweispaltig, seitenweise und nach Namen sortiert ausgegeben und mit einem Rahmen versehen. Weiterhin werden der Name des Datenträgers (Volume-Label), das Erstellungsdatum, die Gesamtgröße in Bytes, der verfügbare Speicherplatz in Bytes, die Anzahl der ausgegebenen Einträge und der belegte Speicherplatz (der ausgegebenen Einträge) angezeigt. Im Gegensatz zum DIR-Befehl zeigt CAT auch versteckte Einträge und Systemdateien an.

Das Programm versucht zunächst, anhand des übergebenen Parameters (ParamStr(1)) und durch Aufruf der Prozedur GetDir, in der Funktion Voller_Pfadname, den vollständigen Suchpfad zu erhalten. Danach sucht es den Volume-Label des gewünschten Laufwerks und bricht ab, falls das Laufwerk nicht bereit ist. Zu beachten ist hier, daß Turbo-Pascal selbständig einen fatalen MS-DOS-Fehler wie „Drive not ready“ abfängt, um dem Programm volle Entscheidungsfreiheit im Fehlerfall zu ermöglichen.

Ein Programm, das ein sortiertes Inhaltsverzeichnis ausgibt und zusätzlich auch gleich noch die MS-DOS-Dateiattribute anzeigt, ist nicht nur eine wertvolle Hilfe: Es ist die Alternative zum DIR-Befehl. Gleichzeitig lernen Sie einige Elemente von Turbo-Pascal 4.0 kennen.

Die Codenummer des aufgetretenen Fehlers befindet sich nach Ausführung der jeweiligen DOS-Routine in der vordefinierten Variablen DosError.

Anschließend sucht das Programm alle Einträge, die auf die Maske des Suchpfades zutreffen und sortiert die entstandene Liste mit dem bekannten Quicksort-Algorithmus. Die Daten werden dabei in einem Feld des Typs „Array [1..max] of SearchRec“ gespeichert. Dieser Variablentyp (*Bild 3*) wird standardmäßig in der Unit DOS deklariert. Die Variable n enthält die Anzahl der gefundenen Einträge. Das Programm erzeugt nun die zweispaltige umrahmte Liste. Jeder Eintrag

```
{ $DEFINE MITCRT }

***
{ $IFDEF MITCRT } crt, { $ENDIF }
dos;

const
  max=          1000;
  InfoBreite=    36;

var
  Liste:         array[1..max] of SearchRec;
  DirInfo, VolumeInfo: SearchRec;
  n:             word;
  Pfad:          string;

procedure Sort_Liste;
var
  x, y: SearchRec;

procedure qsort(l,r: integer);
var
  i,j: integer;
begin
  i:= 1;
  j:= r;
  x:= Liste[(l+r) DIV 2];
  repeat
    while Liste[i].Name < x.Name do inc(i);
    while x.Name < Liste[j].Name do dec(j);
    if i <= j then begin
      y:= Liste[i]; Liste[i]:= Liste[j]; Liste[j]:= y;
      inc(i);
      dec(j);
    end;
  until i > j;
  if l < j then qsort(l,j);
  if i < r then qsort(i,r);
end;
```

Bild 1. Ein Pascal-Programm als Alternative zum DIR-Befehl

```

begin
  if n > 0 then qsort(1,n);
end;

function RectToOutput(r: SearchRec): string;
const
  Leer: string[30] = '          '; {30 x}
var
  t, s: string;
  l: byte absolute t;
  p: word;
  dattim: DateTime;
begin
  if r.Attr = Directory then begin
    t := '          ';
    t[0] := #0;
    t := ' ' + r.Name;
    t[0] := #26;
  end
  else if r.Attr = VolumeID then begin
    t := ' ' + r.Name + ' vom ';
  end
  else begin
    {--- Vorname ---}
    p := pos('.', r.Name);
    if p = 0 then p := length(r.Name) + 1;
    t := '      .';
    t[0] := #0;
    t := ' ' + copy(r.Name, 1, p-1);
    t[0] := #10;
    {--- Nachname ---}
    t := t + copy(r.Name, p+1, 3);
    t[0] := #13;
    {--- Größe ---}
    str(r.Size:8, s);
    t := t + s + ' ';
    {--- Attribute ---}
    if r.Attr and ReadOnly = 0 then t := t + ' ' else t := t + 'r';
    if r.Attr and Hidden = 0 then t := t + ' ' else t := t + 'h';
    if r.Attr and SysFile = 0 then t := t + ' ' else t := t + 's';
    t := t + ' ';
  end;
  {--- Last Modify: Date ---}
  UnPackTime(r.Time, dattim);
  str(dattim.Day:2, s);
  t := t + s + ' ';
  str(dattim.Month, s);
  if length(s) = 1 then s := '0' + s;
  t := t + s + ' ';
  str(dattim.Year:4, s);
  t := t + s;
end;

if r.Attr = VolumeID then
  RectToOutput := t
else
  RectToOutput := copy(t + Leer, 1, InfoBreite);
end;

procedure Liste_Ausgeben;
const
  maxlen = 19;
  sRechts: string[3] = '  |';
  sLinks: string[1] = '|';
var
  i, a, e, len: word;
  Belegt: longint;
  { $IFDEF MITCRT }
  c: char;
  { $ENDIF }
procedure zeile_out(l, ml, m, mr, r: char);
var
  i: byte;
begin
  write(l);
  for i := 1 to length(sLinks) + InfoBreite + length(sRechts) - 2 do
    write(ml);
  write(m);
  for i := 1 to InfoBreite + length(sRechts) - 1 do
    write(mr);
  writeln(r);
end;

begin
  if n = 0 then
    writeln('Nichts.')
  else begin
    {--- 2 spaltige Ausgabe ---}
    e := 0;
    repeat
      a := e + 1;
      if (n - a + 1) div 2 > maxlen then
        e := a + 2 * maxlen + 1
      else
        e := n;
      len := (e - a) div 2;
      zeile_out('r', '-', 'r', '-', 'r', '-', 'r');
      for i := a to a + len do begin
        write(sLinks, RectToOutput(Liste[i]), sRechts);
        if len + i + 1 <= n then
          writeln(RectToOutput(Liste[len + i + 1]), sRechts)
        else
          writeln('': InfoBreite, sRechts);
      end;
    end;
  end;
end;

```

```

zeile_out('t','-', 'A', '-', 'J');

if e < n then begin
  write('Fortsetzung ...');
  {SIFDEF MITCORT}
  c:= readkey;
  if c = #0 then c:= readkey;
  writeln;
  {SELSE}
  readln;
  {SENDIF}
end;
until e = n;
{--- 'M Einträge belegen BELEGT bytes' ---}
Belegt:= 0;
for i:= 1 to n do Belegt:= Belegt + Liste[i].Size;
write(n, ' Eintr');
if n > 1 then
  write('äge belegen ');
else
  write('ag belegt ');
write(Belegt, ' bytes');
end;
end;

procedure Speicherverhaeltnis_Ausgeben(lws: string);
var
  lw: byte;
begin
  if copy(lws,2,1) = ':' then
    lw:= ord(upcase(lws[1])) - ord('0')
  else
    lw:= 0;
  write(DiskFree(lw), ' / ', DiskSize(lw), ' bytes frei');
end;

function Voller_Pfadname(p: string): string;
var
  lw, org: byte;
  Pfad: string;
begin
  if copy(p,2,1) = ':' then
    lw:= ord(upcase(p[1])) - ord('0')
  else
    lw:= 0;
  GetDir(lw, Pfad);
  if Pfad[length(Pfad)] <> '\' then
    Pfad:= Pfad + '\';
  if vergist Turbo Pascal
  if copy(p,2,1) <> ':' then p:= copy(Pfad,1,2) + p;
  if pos('\',p) = 0 then
    p:= copy(p,1,2) + copy(Pfad,3,255) + copy(p,3,255);
  Voller_Pfadname:= p;
end;

begin
  {--- Suchpfad herausfinden ---}
  if paramcount = 0 then Pfad:= '.*' else Pfad:= paramstr(1);
  if Pfad[length(Pfad)] in ['\','.',':'] then Pfad:= Pfad + '.*';
  for n:= 1 to length(Pfad) do Pfad[n]:= upcase(Pfad[n]);
  Pfad:= Voller_Pfadname(Pfad);

  {--- Volume ID finden ---}
  FindFirst(copy(Pfad,1,2)+'\*..*', VolumeID, VolumeInfo);
  case DosError of
    0: write('M'J'Volume ist', RectoOutput(VolumeInfo));
    18: write('M'J'Ohne Volumenname');
    else begin
      writeln('M'J'Laufwerk ist nicht ansprechbar. ');
      halt(DosError);
    end;
  end;

  {--- Vollständige Suchpfadangabe ---}
  writeln(' ', Suchpfad ist ' ', Pfad);

  {--- Ist es ein Directory? ---}
  FindFirst(Pfad + '\*..*', Directory, DirInfo);
  if DosError = 0 then Pfad:= Pfad + '\*..*';

  {--- Einträge finden ---}
  FindFirst(Pfad, ReadOnly+Hidden+SysFile+Directory+Archive,
    DirInfo);
  n:= 0;
  if DosError = 0 then
    repeat
      inc(n);
      Liste[n]:= DirInfo;
      FindNext(DirInfo);
    until DosError <> 0;

  {--- Sortieren ---}
  Sort_Liste;

  {--- Ausgeben ---}
  Liste_Ausgeben;

  {--- Speicherbelegung ---}
  write(' ');
  Speicherverhaeltnis_Ausgeben(copy(Pfad,1,2));
  write(' ');
end.

```

Bild 2. Viele Informationen stecken im kompakten Inhaltsverzeichnis

Ohne Volumenname, Suchpfad ist E:*.*

ANSI	.SYS	1651	22.04.1985	CAT	.PAS	6231 r	1.01.1988
AUTOEXEC	.BAT	115	11.03.1988	COMMAND	.COM	22677	15.05.1985
CAT	.DOC	7935 r	1.01.1988	CONFIG	.SYS	88	29.12.1987
CAT	.EXE	11168 r	1.01.1988	VDISK	.SYS	2812	1.11.1985

8 Einträge belegen 52677 bytes (2014208 / 2071552 bytes frei)

im Feld Liste wird in einen String mit der Länge InfoBreite umgewandelt. In Bild 4 ist N der Platzhalter des Dateinamens, x steht für die Größe der Datei, r = Readonly -, h = Hidden -, s = SysFile - Attribut und TT.MM.JJJJ steht für das Datum der letzten Änderung der Datei.

Es wird aus der Variable Liste[x].Time (Typ longint) durch Aufruf der Prozedur UnpackTime gewonnen und hat den Typ DateTime, der im Interface - Teil der Unit DOS deklariert wird.

Die Routinen qsort, RecToOutput, Speicherverhaeltnis_Ausgeben und Volier_Pfadname sind sehr allgemein gehalten und können deshalb in jedes andere Pascal-Programm eingebunden und ohne großen Aufwand benutzt werden.

Bedingte Übersetzung

Das Programm bedient sich in der abgedruckten Version der Unit CRT, die automatisch schnelle Bildschirmausgaberroutinen in das fertige Programm integriert. Löscht man den bedingten Compilerbefehl {\$DEFINE MITCRT}, wird die Unit CRT nicht mehr automatisch eingebunden. Dies wird durch einige im Programm verwendete bedingte Compilerbefehle realisiert. Mit ihnen lassen sich sogenannte Symbolnamen definieren. Anhand der definierten oder nicht definierten Symbolnamen werden jetzt durch Setzen von weiteren bedingten Compilerbefehlen nur die Teile eines Programms kompiliert, die wirklich für bestimmte Entwicklungs- oder Programmversionen nötig sind. Ein Programm (also eine Source-Datei) kann mit dieser Methode zu unterschiedlichen Codes übersetzt werden. Der Programmierer muß nur die Definitionen der Symbolnamen wie gewünscht definieren. Bild 5 zeigt eine Liste aller bedingten Compilerbefehle von Turbo-Pascal 4.0.

Ist im Programm CAT der Symbolname MITCRT definiert ({\$DEFINE MITCRT}), dann übersetzt der Compiler die folgende uses-Deklaration aufgrund der gesetzten Compilerbefehle IFDEF und ENDIF wie folgt:

```
uses
  crt, dos;
```

```
SearchRec = record
  Fill: array[1..21] of byte; { von DOS reserviert }
  Attr: byte;                 { Dateiattribute }
  Time: longint;               { Datum/Uhrzeit (gepackt) }
  Size: longint;               { Größe in Bytes }
  Name: string[12];            { Name }
end;

DateTime = record
  Year,Month,                  { Jahr: 1980..2099, Monat: 1..12 }
  Day,Hour,                    { Tag: 1..31, Stunde: 0..23 }
  Min,Sec: word;               { Minute: 0..59, Sekunde: 0..59 }
end;
```

Bild 3. Die in der Unit DOS deklarierten Typen SearchRec und DateTime

```
Directories: NNNNNNNN===== TT.MM.JJJJ
Dateien:      NNNNNNNN.NNN xxxxxxx rhs TT.MM.JJJJ
VolumeID:     NNNNNNNN TT.MM.JJJJ
```

Bild 4. Das Ergebnis der Funktion RecToOutput

{\$DEFINE SYMBOL}	Definiert den Symbolnamen SYMBOL
{\$UNDEF SYMBOL}	Macht die Definition von SYMBOL wieder rückgängig.
{\$IFDEF SYMBOL}	Kompiliert den folgenden Codeblock nur, wenn SYMBOL definiert ist.
{\$IFNDEF SYMBOL}	Kompiliert den folgenden Codeblock nur, wenn SYMBOL nicht definiert ist.
{\$IFOPT OPTION}	Kompiliert den folgenden Codeblock nur, wenn der Compilerschalter OPTION (B, D, F, I, L, N, R, S, T oder V) wie hinter OPTION (mit + oder -) gesetzt ist. Beispiel: {\$IFOPT I-}.
{\$ELSE}	Kompiliert den folgenden Codeblock nur, wenn die vorangegangene Bedingung {\$IFXXX YYY} nicht zutrif.
{\$ENDIF}	Schließt einen Codeblock ab.

Bild 5. Die bedingten Compiler-Befehle von Turbo-Pascal 4.0

Ist MITCRT undefiniert, dann wird die uses-Deklaration so übersetzt:

```
uses
  dos;
```

Damit erspart sich der Linker das Einbinden von knapp 2 KByte Code aus der Unit CRT. Die Programmausführung verlangt sich allerdings etwas, da die Bildschirmausgaben nun über die langsameren DOS-Interrupts ausgeführt werden.

Dem Programmierer eröffnen die jetzigen Möglichkeiten von Turbo-Pascal völlig neue Wege zur komfortablen Programmentwicklung und -optimierung, sowie zur Erstellung portabler Programme.

Literatur:

- [1] Heimsoeth & Borland: Turbo-Pascal 4.0, Band I und II, 1987.
- [2] Microsoft: MS-DOS 3.1 Programmer's Reference Manual, 1986.

Ulrich Kruppe

Kompatibel, aber schneller

Foxbase+, eine Alternative zu dBase

Mit dem Programm Vulcan später dBase – legte Ashton Tate 1981 so etwas wie den Quasistandard für Datenbanken auf PCs fest: Selbst Konkurrenzprodukte wie Borlands „Paradox“ lesen und schreiben heute ihre Dateien auf Anforderung im dBase-Format. Foxbase geht hier noch einen Schritt weiter: Es benimmt sich einfach genau wie dBase, nur eben mehr als doppelt so schnell.

Im Ursprungsland USA hat sich Foxbase damit schon einen recht guten Platz in den Software-Hitlisten erkämpft. Die deutsche Version (FoxBASE+ 2.0), die diesem Bericht zugrunde liegt, soll diesen Erfolg auch hier möglich machen. Dabei wurden nicht nur die Handbücher, Hilfstexte usw. ins deutsche übertragen, sondern z. B. auch die Funktionen für Groß-/Kleinschreibung auf Umlaute ausgedehnt. Zwar hält die Übersetzung nicht an jeder Stelle der Prüfung meines Deutschlehrers stand („Der Befehl WAIT stoppt eine FoxBASE+-Operation, bis ein Zeichen von der Tastatur erfolgt.“), ist aber immer eindeutig und gut verständlich. Unter der Übertragung in unsere Sprache hat allerdings (besonders beim gegenwärtigen Dollar-Kurs) der Preis ein wenig gelitten: Mit gut 1100 DM ist der Abstand zu dBase-III-plus zwar noch vorhanden, aber in englisch ist Foxbase+ bei Direkt-Importeuren schon für rund die Hälfte zu haben. Trotzdem sollte man sich den Schritt dorthin gut überlegen: Gerade bei einem solch komplexen Programm ist eine deutsche Anleitung, ein deutschsprachiger Ansprechpartner und ein Update-Service für spätere Versionen selbst für den Privatanwender durchaus sein Geld wert, für den professionellen Einsatz sind diese Punkte unverzichtbar.

Kompatibilität

Ein paar kleine Inkompatibilitäten fallen gleich beim Durchsehen der Handbücher auf: Der Befehl „Assist“, mit dem der Anfänger bei dBase-III-plus seine Datenbank menügesteuert aufbaut, ohne die dBase-eigene Programmiersprache lernen zu müssen – diesen Befehl gibt es unter Foxbase

Eigentlich sind im deutschen Sprachraum Flüchse eher ein Symbol für Klugheit denn für Schnelligkeit. So müßte Foxbase hierzulande vielleicht eher „Wieselbase“ heißen, denn klüger (komfortabler) als Vorbild dBase ist dieses Datenbank-Verwaltungsprogramm nicht – wohl aber bedeutend schneller!

ebenso wenig wie die Möglichkeit, Bildschirmmasken (CREATE bzw. MODIFY SCREEN) zu verwenden. Das sind jedoch keine Fehler, mit denen man nicht leben könnte, denn für Foxbase entscheidet man sich ja auch und gerade der Performance wegen. Und die kann man sowieso nur mit einem Programm auf der „Punktoberfläche“, also dem Kommando-Interpreter, wirklich ausnutzen. Dem Fehlen der Screen-Dateien läßt sich bei Bedarf mit einem preiswerten Maskengenerator (z. B. „Saywhat“ für 150 DM) abhelfen.

Außerdem wird man auf folgende Abweichungen zu dBase-III-plus hingewiesen: Variablenamen dürfen bei Foxbase+ nicht länger als elf Buchstaben sein und – wie Feldnamen – keine Umlaute enthalten; die Befehle IMPORT, EXPORT und CREATE/MODIFY QUERY werden nicht unterstützt. Auf der anderen Seite bietet Foxbase+ eine ganze Reihe – das Handbuch spricht von rund 40 – zusätzlichen Funktionen, vor deren Verwendung man sich jedoch ein paar grundlegende Gedanken bezüglich Übertragbarkeit von Programmen machen sollte. Ansonsten aber gibt es keinen Grund zur Klage, denn es laufen alle dBase-Programme auf Anhieb. Mehr noch: Die bei dBase in Listen aufgeführten „bekannten Fehler“ gibt es bei Foxbase nicht – hier werden Fehler bei Bekanntwerden beseitigt, die Befehle tun also immer, was man von Ihnen erwartet.

Die Steigerung der Geschwindigkeit gegenüber dem Vorbild erreicht Foxbase unter Anderem durch eine konsequente, sinnvolle Nutzung des RAM. Das hat zwar (wie jede glänzende Medaille) seine Kehrseite – so läßt sich etwa über Foxbase mit einer großen Datenbank noch ein COMMAND.COM laden, der hat dann aber keinen Platz mehr, um noch irgend ein Programm zur Ausführung zu bringen – aber es macht

auch einen wirklich beeindruckenden Geschwindigkeitsvorteil aus. Außerdem bringt diese Methode jeden Tester in arge Bedrängnis: Wie groß der Geschwindigkeitsvorteil im Einzelfall ist, hängt damit nämlich nicht nur von der Größe der Datenbank und der ausgeführten Funktion, sondern auch

vom freien Speicherplatz, von Festplattenzugriff und -fragmentierung und ähnlichen unwägbaren Parametern ab. Mit der Angabe „mehr als doppelt so schnell“ liegt man aber bei den für eine Datenbank wesentlichen Funktionen immer auf der sicheren Seite.

Foxbase ist kein Interpreter, sondern verarbeitet seine Programme in einem Zwischencode. Dieser Zwischencode wird entweder beim Laden aus einem Quelltext erzeugt (was lange Ladezeiten verursacht, aber schnelle Ablaufzeiten garantiert) oder durch den mitgelieferten Compiler schon vorher generiert. Das Laden und der Ablauf solcher vorcompilierter Programme ist natürlich gegenüber jedem reinen Interpreter erheblich schneller.

Derart compilierte Programme laufen auch mit dem „Runtime“-Paket, das man (für einen dem Foxbase+-System ähnlichen Preis) erstehen und zusammen mit seinen Datenbank-Programmen in unbegrenzter Stückzahl weitergeben kann. Firmen brauchen also keine große Zahl von Lizenzen: Ein „Entwicklungs“- und ein „Runtime“-Paket, das reicht.

Zur Orientierung: Ein compiliertes Datenbank-Programm mit rund 500 Zeilen ist 15 Sekunden nach dem Aufruf von der Festplatte eines Standard-PC (8088 mit 4,77 MHz) bereit zur Ausführung, wenn die gleichzeitig geladene Datenbank etwa 1000 Einträge hat. Das erfolgreiche Durchsuchen dieser Einträge ist in rund zehn Sekunden abgeschlossen.

Für den, der heute eine Datenbank mit einem gutem Preis-/Leistungsverhältnis braucht und weder auf die Unterstützung durch das Software-Haus noch auf Software-Werkzeuge (FoxToolBox, Saywhat usw.) verzichten will, für den ist Foxbase+ ein Kandidat, der ganz oben auf der Liste in Frage kommender Programme stehen sollte.

mc-quickies sind aktuelle Produktanzeigen, mit denen Firmen ihre Produkte vorstellen. Verantwortlich für den Inhalt sind die Inserenten.

Trans-Modem 2400 698,-
CCITT V.21/V.22/V.22bis 300/1200/2400 Baud

Modemkarte PC 1200 298,-
Kurze PC-Karte, V21/22, 300/1200 Bd voll duplex,
RS 232 eingebaut (COM 1-4), Lautsprecher.

KS 1200, extern, sonst wie PC 1200 368,-
Pocketmodem 1200 398,-

Alle Modems Hayes-komp. und ohne FTZ-Nr.
Anschluß an das Postnetz nicht erlaubt.

RAMboard 2 MB 349,-
EMS-/Intel-komp., ohne RAM, mit Software

AT-RAMboard mit 16-Bit-Slot, sonst w. o. 449,-

AD/DA-Wandler 269,-
Auflösung 12 Bit, 16 AD-, 1 DA-Kanal

VIA 8255 179,-
I/O-Karte, 48 Leitungen, 16 LED, Timer/Counter

Netzsicherung PT 650 1995,-
Belastung bis 550 VA, 15 Min. bei Halblast

ARClick-Netzwerkkarte 598,-
Z. B. für Novell, Info über weitere
ARClick-Produkte anfordern

bsb
Hohenzollerling 74, 5000 KÖln 1
Telefon 02 21-13 14 41

mc-AllPro ^{Plus}



**Universelles Programmiergerät aus mc 6-8/1987
mit erweiterter Software**

- **Programmiert jetzt auch GALS**
- **PROBs von TI und Zeropower RAMs**
- **Vorbereitet für Single Chip Prozessoren von INTEL**
- **Unterstützt das Intel Hex Format**
- **Inhaltsverzeichnisangeige**

- Programmier (EEPROMs bis 27612, 27613 und 2701) PROMs von Signetics Vahco und PALs von MMI, MS, TI und AMD
- Anschluss über RS 232 (V 24) Schnittstelle
- Hochleistungs-Turbo ROM 114 XT AT (unter PC/MS DOS) kompatibel und Z80 Systeme (unter CP/M) verfügbar
- Mikroformantensteuer (EEPROM-Editör)
- Mikroformantensteuer Test Editor zum PAL-Entwurf
- PAL-Assembler + PAL-Simulator + MS-DOS-Assembler
- Funktionen als JED-Dateien speicherbar

/ Bausatz (mit Handbuch und Software) 997,- DM

/ Fertigplatine (geliefert Ready Software) .. 1420,- DM

Versand per Nachnahme. Bei Bestellung unbedingt Zahlungsreihe, Dienstleistungsform und Betriebsversion angeben!

Holger Haase & Michael Menrad
Peripheriegeräte GbR
Postfach 2847 / 3300 Braunschweig
Telefon (0531) 400310

INTERFACES

XON/XOFF ↔ RTS/CTS

- Datenübertragungstester
- Galvanische Trenn-Adapter (1000V)
- High-Speed-Adapter (10 MB/s)
- Störsichere Übertrager
- Long-distance-Interfaces (20 km)
- Datenformat-Wandler (Asynchron)
- Baudrate Konverter
- seriell ↔ parallel-Wandler
- Kurzstreckenmodems > 100 Kbaud
- Netzwerk-Adapter
- Microprozessorgesteuerte Interfaces
- Mehrweg Druckpuffer 256 KB
- Kundenspezifische Interfaces

RS232C (V24 V28)	RS423 (V 10)
Current Loop	TTY
(20 40 60mA)	CENTRONICS
RS422 (V11) RS485	IEEE 488 IEC HP IB
PIN 1360 (X.27 TR30 1)	BCD

Dipling Jürgen Knauth

• 2. u. 3. Stockwerk • Industriestraße 1 • 40699 Düsseldorf • Tel. 0211 4611-1

MSCT 7




Seit fast zwei Jahrzehnten vom Videobereich TV, Home und Recorder bis in die Gebiete der Computertechnik für IBM AT & XT

VIDEOK-1000 I 405.- DM
Auflösung 640 x 320 und 640 x 256
Preis nur 32 Geführter Lernstufen!
64k 640 x 320 und 320 x 320
RGB 640 x 320 und 320 Pixel
64k 640 x 320 und 640 x 320

VIDEOK-1000 II 796.- DM
Auflösung 1280 x 1024 und 640 x 640
Preis nur 32 Geführte Lernstufen!
64k 640 x 320 und 320 x 320
RGB 1280 x 1024 und 320 Pixel
64k 640 x 320 und 640 x 320

UTILITIES 1 99.- DM
 1000-Programme aus 36 Programmen. Kann man AT, XT, PC, IBM, Dos, Windows, Drucken, Mailbox, Programmieren, Diskkette, etc. etc. Programmieren, testen, etc. Bitte bei Bestellung VIDEOK 1000 I oder II angeben.

UTILITIES 2 99.- DM
 1000-Programme aus 36 Programmen. Kann man AT, XT, PC, IBM, Dos, Windows, Drucken, Mailbox, Programmieren, Diskkette, etc. etc. Programmieren, testen, etc. Bitte bei Bestellung VIDEOK 1000 I oder II angeben.

Infobrief gratis (kostenlos) - falls Sie geringere Einzahlung von 30 DM leisten oder Bestellungen über Versand an: **Computer-Service GmbH**

Ing. Bodo Manfred Fricke **Reise 500 13** **1000 Serie 37** **795 030 001 50 52**

[illegible]

C-COMPILER

MI-C für CP/M, CP/M 86, MS-DOS

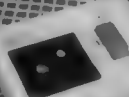
**MI-C vereint hohen Bedienungskomfort
mit hervorragender Leistung**

- Vollständige Version mit 13stelliger BCD-Arithmetik für Scientific Computing
- Erzeugt kurze und schnelle ROM-fähige Programme
- Ausgabe im Assemblercode des Zielprozessors
- Kompatibel zu MAC80/L80 (MASM/LINK) v. Microsoft
- Für 8086: 4 Speichermodelle/8087-Co-Prozessor
- Fehlerverfolgung mittels Trace möglich
- Umfangreiche Bibliothek (AMD9511-Paket erhältlich)
- UNIX-kompatibel
- Deutsche oder englische Version lieferbar

8"-/5,25"/-3,5"/-3"-Disk + dt. Handbuch	
MI-C für CP/M	445,- DM
MI-C für CP/M 86 oder MS-DOS	575,- DM
MI-C-Crosscompiler (Ziel 8080/286)	745,- DM
MI-C-Crossassembler + Linker (Ziel 8080/286)	845,- DM
MI-C-Crosscompiler/-assembler (Ziel 8051)	1.495,- DM
MI-C-Crosscompiler/-assembler (Ziel 8096)	1.495,- DM
MI-C-Crossassembler (Ziel 68000/10/20)	795,- DM
MI-C-Crossassembler + Linker (Ziel Z280)	795,- DM

Herbert Rose, Bogenstraße 32, 4390 Gladbeck,
Telefon 0 20 43/2 49 12 und 4 35 97

Vertrieb in Österreich:
Dr. Wilibald Kraml, Microcomputer-Software,
Degengasse 27/18, A-1160 Wien



EPROM - Brenner
für 2716 · 27512/513

Neue Software V3.0
Auto-Setup & Window-Technik

Dieses professionelle Programm ermöglicht für Entwicklung, Service, Produktion und Schulung Kunden-Frist und angeregten Austausch. Anschluß an die Centronics-Druckerschnittstelle ist möglich. Keine Steckkarte im Rechner erforderlich! Das Programm wird in 1/8" Vorformat. Aufgrund des erweiterten Preis-Leistungsverhältnisses befindet sich dieses nunmehr weit über 1500 Exemplare im täglichen Einsatz in Industrie, Entwicklung, Fachschulen, Schulen und Schölung.

Software V3.0: Für PC/XT/AT und Kompatibles Auto-Setup für automatisches Anpassen des Programmsystems an Ihre vorhandene Hardware. Grafikadapter, Betriebssysteme, Laufwerke, Software, Windows, Fonts, Interrupts, unterstützte Auswahlen sind im "Systemeditor" einsehbar, nur aus Systemeditor integrieren sie Änderungen ins System und bestätigen es mit einer Speicheradresse. Die Daten werden automatisch gespeichert und wieder geladen d.h. 27512/513 sind in einem Durchgang editier-, les- und programmierbar. Integrierter komfortabler "Fullscreen-Editor" für ASCII u. HEX, 16-Bit breite Programmierung (Normal-Low-High Byte). Österreichische INTEL-Matrise, exportiert.

• Platte pro V2.01, Handbuch und Software pro V2.11	DM 29,00
• Bezaugt pro V2.01, Handbuch und Software pro V2.11 (ohne Gehäuse)	DM 24,00
• Programmgerät pro V2.0 incl. Software pro V3.0 und Handbuch	DM 245,00
• Software pro V2.11 incl. Quickcode für PC/XT/AT und Kompatibles	DM 29,00
• Software pro V3.0	DM 68,00
• Software-Gehäuse (unbearbeitet wie Bild Größe: 180x140x75 mm)	DM 29,00
• Druckerbauteil für XT/AT und Kompatibles	DM 29,00

Versandkosten Ausland DM 12,00 Inland DM 6,00 Versand per Nachnahme

B & P

Beate Sued Ludwig Durr-Str.10 7320 Göppingen Tel. 07161-73241

Z-80-EURO-KIT

bestehend aus:

- 1 Macro-Assembler Z-80 für IBM
- 1 Eprommer für IBM
- 1 Z-80-Euro-Rechner
- 1 Z-80-Buch zur Einführung

komplett: 1599,— DM

Was bisher nur ein Traum war, wird nun WIRKLICHKEIT! Programmieren Sie auf Ihrem IBM-PC mit unserem Z-80-MACRO-ASSEMBLER in den bekannten Mnemonics, legen Sie den Z-80-SOURCE in einem EPROM ab und probieren Sie Ihr Programm an unserem vielseitigen Z-80-Rechner (EUROFORMAT) aus. Das Buch wird Ihnen dabei helfen.

KOSTENLOSE INFO anfordern. Versand erfolgt per NN oder Vorschek. EXPORT ist gegen Vorschek möglich.



Computer Ring

Heinrich Kotter Electronic
Steinstraße 22, 5042 Erftstadt
Postf. 11 27, Tel. 0 22 35/7 67 07
Fax 0 22 35/7 20 48



VME-Bus + ATARI ST

Die anwenderfreundlichen Systeme für

**Meßdatenerfassung
Prozeßkontrolle
Automatisierung**

Hoher Bedienerkomfort, einfache Programmierung und günstige Preise ermöglichen auch die Lösung Ihres speziellen Problems.

Fragen Sie uns!

rhothron GmbH

Rudolfstr. 14 · 7500 Karlsruhe · Tel. 07 21/6 03 11

mc-MINIMARKT

Einige gebrauchte CBM-8032-SK-Rechner à DM 320.- und einige zerlegte CBM-8032-Rechner à DM 240.-.

ETech. Büro, ☎ 06652/505-0

MC68000, OS/9-Betriebssystem, 512 KByte RAM, 512 KByte CMOS-RAM für OS/9, IBM-Tastatur u. Gehäuse, grüner Monitor, DM 1200.-.

☎ 0 40/6 41 10 39, abends

MC68010-Motherboard, Floppy-Kontrollern, SASI-Interface, Terminierung, NT 2, Gehäuse und Dokumentation, funktionsgeprüft durch DSM, VB 800.-.

☎ 04 21/87 69 92

Typenraddrucker Brother HR40, Einzelblatteinzug, und Endlosbetrieb, neuw., VB 1700.-.

☎ 0 70 42/2 43 88

Gebr. 128er Epr. à 5.-, Schalt- netzteile, 250 W 230.-, 70 W 139.-. ☎ 0 95 21/89 63, ab 17 Uhr

SERVICE-MANUALS für FX 80, CP 80, CPA 80, C 64 sowie alle TAXAN-Geräte (gebunden) jeweils DM 50.-. Regge, Fesenfeld 57, 2800 Bremen 1, ☎ 04 21/7 11 14

MC-CP/M-Rechner, Preis VB.

☎ 0 60 07/28 43, ab 18 Uhr

Tausche Regel-Trenntrafo 2000 VA/8 A/0-280 V, neuw., gegen Ossi. ab 20 MHz, 2- oder 3strahl. ☎ 04 51/80 22 46, ab 18 Uhr

NDR-Computer, Z80H, SBC3, ROA64, RAM256, BUS3, NE1, IOE, CAS, FLO3, KEY, CENT, GEH3, TAST, MONITOR, EGRUND, EFLOMON, EZEAT, EBASIC, EGOSI, ESKOP, LOOP1-16, MC1+2, Literatur, VB 700.-.

☎ 0 40/2 99 12 59, ab 18 Uhr

NDR-Computer-Einstiegspaket mit Z80-Buch, Kleinbuch und GRAF-Katalog zu verk.

☎ 0 62 02/1 65 13

Typenraddrucker CBM 8028 200.- Papiereinzug BDT ASF176 180.- für bel. Drucker. ☎ 0 81 57/70 91

EGA-Karte (ATI-Wonder) 330.-.

☎ 0 61 61/28 53, ab 18 Uhr

MC-CPM-System, in Gehäuse, SYS1 OUT 1, FLO1, TERM1, 2x5¼"-Laufwerk, Tastatur, 40 Disketten, Unterlagen, VB 9000.- öS. Wien, ☎ 02 22/ 9 48 96 35, ab 17 Uhr

MC68000, 512 KByte RAM, 2x TEAC, Tastatur, Monitor, CPM-68K, OS-9. Preis: VHS.

☎ 0 72 43/1 66 01

NDR-Comp., 19"-Geh., Z80 u. 68008, 15 Karten, gr. Tast., Monitor, viel Softw. u. Literatur.

☎ 0 71 62/39 98

MC68000, 1,5 MByte, 2 LW, Monitor, Tastatur, Gehäuse, EProm-Karte OS9, CPM68K M-Disk, Gred Xedet Basic usw., DM 2500.- VB. ☎ 0 81 23/7 00

RAM-Chips 256 KByte 100 ns, 256 KByte 60 ns und 1 MByte 100 ns, günstig abzugeben.

☎ 0 53 71/44 54

14"-Monitor-Chassis (P31/P39), Hercules-kompatibel, nur 59 DM. Gerloff-Elektronik, ☎ 0 51 46/86 81

kws-SAM 68k/640-10H Einplatinencomputer, 10 MHz/640 KByte RAM inkl. Handbuch und Rückwandmodul RTC 1 (Neupreis TDM 3 netto), VB DM 1700.-. IVC der HBK, Postf. 28 28, 3300 Braunschweig

SOFORTDIENST!

CAD-Layouts, Leiterplatten, Frontplattendruck in High-Industriequalität. AS-Marketing, Schnackenburg-Allee 14, 2000 Hamburg 54, ☎ 040/85082 62

68000 Asm. Realzahl FFT/S Atari VOID C: in GfA-Basic, DM 150.-. Auf Wunsch sind optim. Vers. für spez. 2N verfügbar.

Zuschr. unter MC Nr. 108

Preisrenner! 80286-Speedkarte-XT, 10 MHz, DM 479.-; 80286 Baby AT, 20 MByte HDD, 1,2 MByte FDD, 512 KByte, 102 Key-Tastatur, DM 2698.-; LCD-Portable-286, 20 MByte HDD, 512 KByte, 1,2 MByte FDD, 80 Zeichen LCD, DM 3448.-. Conical Computer, Lippeltstr. 1, 2000 Hamburg 1, ☎ 0 40/33 60 34/35

MC68010, 512 KByte RAM, Terminatorkarte, Sprach-/Uhren-Karte, EPROM-/S-RAM-Karte, Floppy 1 MByte, Schaltnetzteil, Gehäuse, gr. ELZET-Tastatur, Monitor mit Schwenkfuß, Software plus Unterlagen, DM 1000.-. 2 MByte RAM DM 500.-. ☎ 0 80 92/97 55

.....
68020/881-Huckepack inkl. Softw. für MC68000, Log. o. Proz. sFr. 270; stat. RAM 256 KByte bestückt sFr. 420; TERMINALDRIVER OS9 (virt. Term. div. Funktionen sFr. 100; FILETRANSFER OS9-CP/M sFr. 80; ab Lager bei STEINMANN & FREI, Postf. 565, CH-8820 Wädenswil

.....

Verkaufe Apple IIc, Aufger. 512 KByte, kompl. mit ext. Laufwerk FD1 u. Tragtasche, alles original-verp. für 1750.- DM.

☎ 0 95 33/7 83

● Kamera für Osci und Monitor ● Laborwagen

Traumhafte Preise z. B.

- D. Multimeter ab DM 108.-
- D. Multimeter TRUE RMS 3 Stck ab DM 98.-
- F. Generator ab DM 450.-
- P. Generator ab DM 412.-
- Elektron. Zähler ab DM 399.-
- Netzgeräte - jede Preisklasse
- Meßkabel - Testköpfe - R, L, C Dekaden
- Adapter - Stecker - Buchsen
- Video-Audio-Kabel u. v.m.

Prospekt kostenlos Händleranfragen erwünscht

HAMEQ sofort ab Lager **HAMEQ**
Beckmeier electronic · Danziger Str. 4b
2804 Lillanthel · ☎ 0 42 88-46 80
HAMEQ sofort ab Lager **HAMEQ**

Jetzt wieder lieferbar!

Basic für Aufsteiger

Der sichere Weg zum fortgeschrittenen Basic-Programm. Von R. Busch. 3. Aufl., 285 S., 74 Abb., 11 Tab., geb. DM 58,- ISBN 3-7723-7283-X Die meisten Basic-Einführungen hören



nach einer ersten Kontaktnahme mit wenigen Befehlen auf. Wer weitermachen will, der hat das richtige Buch in der Hand. Es vermittelt nicht nur eine Menge an Wissen, es ist auch unterhaltsam zu lesen.

Das Franzis-Fachbuch

Ein Qualitätsversprechen

Franzis-Verlag GmbH, Karlstraße 37-41, 8000 München 2

mc 8-88-24

Das ENDE

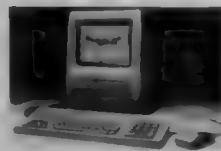


...aller leeren GRAFIK- und BILDER-dateien für Ihr Textverarbeitungs- oder DTP - Programm ist da!

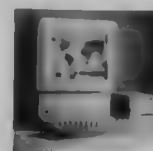
PICTURE DISKS

1000 hochqualitative Grafiken im PC-Format für nur 198,- DM (zzgl. P+V / Zahlung per NN, V-Scheck oder bar)

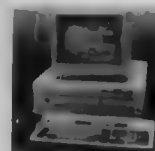
Ingenieurbüro ARNOLD 6749 Birkenhördt Gehlmühle 3 ☎ 06343/3787



Macintosh



Apple II



MS-DOS

HARDWARE SOFTWARE ZUBEHÖR BÜCHER

Spezialkatalog für ☐ Apple II, ☐ Macintosh, ☐ MS-DOS

Fordern Sie unter Angabe Ihres Rechnertyps den entsprechenden Gratiskatalog an!

pandayoft Dr.-Ing. Eden

Uhlandstr. 195 D-1000 Berlin 12

Tel.: 030 / 31 04 24

Telex: 185 859

BITTE SCHICKEN SIE MIR MEIN GRATISKATALOG ZU!
Name: _____ Adr.: _____ Rech. Typ: _____

MC

August 1988

INSERENTENVERZEICHNIS



Die Mikrocomputer-Zeitschrift

REDAKTION

Anschrift:

Franzis-Verlag GmbH
Karlstraße 37-41, 8000 München 2
Postfach 37 01 20, 8000 München 37
Sekretariat: Rita Schieser
Telefon: 0 89/51 17-3 54
Telex: 5 22 301
Telefax: 0 89/51 17-3 79

Chefredakteur:

Dipl.-Math. Ulrich Rohde, verantw.
(Anschrift der Redaktion)

Stellv. Chefredakteur:

Dipl.-Ing. (FH) Ulrich Kruppe

Redaktion: Horst Brand, Dipl.-Ing. (FH) Wolfgang
Hascher (li), Aktuelles: Reiner Schönrock, PC-
Technik: Dipl.-Ing. (FH) Dieter Strauß

Ständige Mitarbeiterin (zu erreichen unter der
Anschrift der Redaktion): Heiga M. Schmidt

Korrespondent:

USA: Ray Duncan

Art Direction: Dipl.-Des. (FH) V. Hilbel

Layout, Grafik, Herstellung:
Josef Würzinger

Labort: Neucom Electronic GmbH

Software-Service:
Shamrock Software-Vertrieb,
Telefon 0 89/59 54 68

Sonderdrucke: Jakob Wintersberger

Lizenzen: Georg Geschke, Königsstr. 8,
3000 Hannover 1, Tel. 05 11/34 53 62

Gesamtherstellung: Franzis-Druck GmbH,
Karlstraße 35, 8000 München 2,
Telefon 0 89/51 17-1

VERLAG

Anschrift:

Franzis-Verlag GmbH
Karlstraße 37-41, 8000 München 2
Postfach 37 01 20, 8000 München 37
Telefon: 0 89/51 17-1
Telex: 5 22 301
Telefax: 0 89/51 17-3 79
Postgütekonto München 57500007

Geschäftsführer:

Peter G. E. Mayer,
Michael-Alexander Mayer,
Dr. Harald Wiebking

Anzeigenleitung: Dieter Köster, verantw.
(Anschrift wie Verlag)

Anzeigenverkaufsführung: Hans-Joachim Hecht
(-3 86)

Disposition: Ingrid Deschner (-2 97)

Vertriebsleitung: Herbert Barnehl

Abonnement: Christa Fischer (-2 40/-2 79)

Handelsverkauf: Dietlind Stauder (-2 04/-2 83)

Bezugspreise Inland: Einzelheft 7,- DM, Jahres-
abonnement 70,- DM; Vierteljahresabonnement
18,50 DM. Studenten, Auszubildende und Ren-
ner erhalten das Jahresabonnement gegen Nach-
weis günstiger. Der Versand ist im Abonne-
mentspreis eingeschlossen. In den Preisen ist die
gesetzliche Mehrwertsteuer in Höhe von 7%
enthalten.

Die MC erscheint monatlich, jeweils montags am
Monatsanfang bzw. am Ende des Vormonats; im
8. Jahrgang.

ISSN 0720-4442

Vertriebskennzeichen B 7745 E



VERLAGSVERTRETUNGEN

Anzeigenvertretung Inland:

Baden-Württemberg: Ulrich G. Felger, Honold-
weg 27, 7000 Stuttgart 1, Tel. 07 11/63 27 18

Bayern: Elise Busch, Münchner Verlagsvertre-
tung, Sperberstraße 23, 8000 München 82, Tel.
0 89/4 30 73 32

Berlin: Rainer W. Stengel, Bischofsgrüner Weg
91, 1000 Berlin 40, Tel. 0 30/7 74 45 16

Hessen: Günter Junne, Victor-Achard-Str. 30,
6380 Bad Homburg v. d. H., Tel. 0 61 72/333 94

Norddeutschland: Lita Lange, Impulse Medien
service GmbH, Holtensklinter Str. 9, 2050 Ham-
burg 80, Tel. 0 40/7 24 20 37/38

Nordrhein-Westfalen: Bandelow + Partner GmbH
& Co. KG, Herr Werner Bandelow, Dellestraße
36, 4000 Düsseldorf 12, Tel. 02 11/20 14 64

Anzeigenvertretung Ausland:

Schweiz: Exportwerbung AG, Kirchgasse 50, CH-
8024 Zürich, phone: 01-47 46 90, telex
8 12 765

Großbritannien: Martin Geerke, 4, Friary Hall
(Flat 3), Friary Road, South Ascot, Berks SL5
9HD, U. K., phone: 9 90-2 86 49, telex:
858 328 EUROAD

Japan: ABC Enterprises Inc., Helox W. Kuhl-
mann, 7-4, Ohayama-cho, Shibuya-ku, Tokyo 151
Japan, Tel. 4 85-29 61-3, Fax 4 66-07 09

USA: DemoNet Inc., 7330 Adams St. Paramount,
CA 90723, phone: (213) 408-0999, telex No.
(910) 321-3026

France: Agence Gustav Elm, 41, Avenue Mon-
taigne, 75008 Paris, phone: 01-47 23 32 67

Italien: Rancati advertising, Milano San Felice Tor-
re 5, I-20090 Segrate, phone: 02-7 53 14 45,
telex: 3 11 250 PPMIL

Belgien: BCL/United Media Int. S.A., Avenue de
la folle chanson, 2 bte 7, 1050 Bruxelles, Tel.
02/6 47 31 90, Telex: 6 3 950 eci um

Anzeigenpreise nach Preisliste Nr. 8, gültig ab
1. 10. 1987

Verlagsvertretungen Ausland (Bezugspreise in
Klammern):

Belgien: Office International des Périodiques
(O.I.P.), Avenue Marinx 30, B-1050 Brüssel (Ein-
zelheft 182,- bfr, Jahresabonnement 1903,- bfr)

Dänemark: Harck + Cjellerup Bookseilers Ltd.,
Fiolstræde 31-33, DK-1171 Kopenhagen K.
(Jahresabonnement 316,- dkr)

Frankreich: Librairie Parisienne de la Radio, 43,
rue de Dunkerque, F-75010 Paris

Luxemburg: Messageries Paul Kraus, 5, rue de
Hollerich, Luxembourg

Niederlande: De Mulderkring BV, Electronics
House, Postbus 313, 1380-AH Weesp (Einzelheft
8,50 hfl, Jahresabonnement 164,50 hfl)

Österreich: Erb-Verlag Ges.m.b.H. & Co., KG,
Buch- u. Zeitschriftenvertrieb, Amerlingstr. 1, A-
1061 Wien (Einzelheft 60,- S, Jahresabonne-
ment 620,- S)

Schweiz: Verlag Thalig AG, CH-6285 Hitzkirch
(Luzern) (Einzelheft 7,- sfr, Jahresabonnement
67,20 sfr, je nach Kursituation)

Urheberrechte: Die in der Zeitschrift veröffent-
lichten Beiträge sind urheberrechtlich geschützt.
Für Bauanleitungen, Schaltungen und Program-
me zeichnen die Verfasser bzw. Entwickler ver-
antwortlich; für Fehler im Text, in Schaltbildern,
Aufbauzeichnungen, Programm-Listings usw. kann die
Redaktion weder eine juristische Verantwortung
noch irgendeine Haftung übernehmen.

Printed in Germany. Imprimé en Allemagne. ©
1988 für alle Beiträge bei Franzis-Verlag GmbH

Abor	11	Knauff	130
AD Computer	23	Kolter	130
Arnold	163	Krischer	139
ASC-Elektronik	132	Kwem	15
A.S.S.-Ware	134		
ATS	161	Laserland	134
		Laytronic	131
B & P	130		
Bockstaller	132	M + H	162
bsb Datentechnik	130	Magtron	133
		Mathes	140
Cherry	8, 9	Mayon	134
Chicony	143	McGraw-Hill	154
Christiani	133	MCI	85, 85, 167
Chung Yu	146	Medinger Electronic ..	131
Computer		Meyer E. W.	11
Discount 2000	163	Micromint	159
Conex	11	Microtech	157
		Milde	131
Datapro	133	Müller, Dr. Gert	132
De Uneo	146	MoVe	134
Dobbertin	131	Multisoft	134
DSM	18, 19		
		NBN	13
EcoSoft	131	Neucom	133
Elco	137		
Elektronikladen	17	Pandasoft	163
Express Service	156	Plantron	2
Franzis	31, 39, 136,	Ranfft	134
	150, 151, 152, 153	Rheintec	147
Fricke	130	Rhotron	130
Fujitsu	37	Rose	130
		RWL	155
GFM	163		
Gorny	133	Seitz	156
Graf	29	Shamrock	119
		Siemens	34, 35
Haase & Menrad	130	Sintronic	138
Habersetzer	152	Super Nature	147
HK electronic	133	Super Smart	145
Holco	144		
		Toshiba	7
Indutronic	3	Trost	21
Intec	132	TSS-Schmitz	132
Interest-Verlag	26		
Intra	59	Ueding	162
iSystem	149		
		Vobis	168
Janus	132	Vogel-Verlag	103
Jeschke	131		
		Walter	139
Kind	161	Wiegand	153
Kirschbaum	16	Wiesemann & Theis ..	158

VORSCHAU

Große Testaktion

Dieter Strauß, unser PC-Spezialist hat seine Benchmark Suite automatisiert und den Herbstmodellen von IBM und Compaq genau auf die Tastatur geschaut. Lesen Sie, was die neuen Modelle leisten und schauen Sie, wie sie aussehen. Kompliziert ist zum Beispiel das Zeitverhalten der VGA-Karten. Wie bei allen Benchmarks gilt: auf die Interpretation kommt es an. Aus Taiwan kommt von der Firma Mitac ein 286er-PC, der es in sich hat.



High Tech

Was nützen die Transputer ohne Software, die Parallelität unterstützt? Inmos hat dafür Occam geschaffen und ein Entwicklungspaket darum herum. Daneben gibt es einige C-Versionen, die Parallelität eingebaut haben. Auch für die mc-Transputerkarte gibt es ein Cross C, das auf dem PC compiliert wird und auf der Transputerkarte läuft. Jetzt auch mit Unterstützung vernetzter Transputer.

IEEE-Mathematik

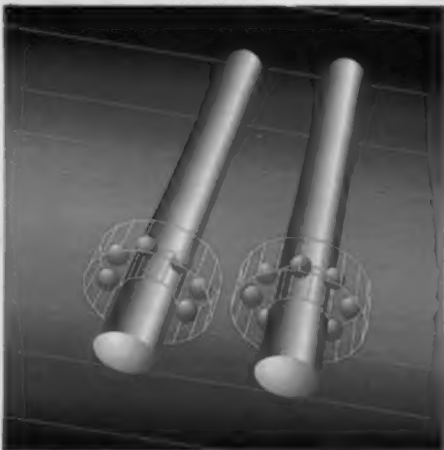
Ab der Ausgabe 9 etwas Grundlegendes über den Standard in Sachen Gleitkomma, den die IEEE-Organisation gelegt hat. Schritt für Schritt werden die Grundla-



DOS- und C-Workshop

Zum Beispiel ein Programm für C-Fans, das aus .C-Dateien eine .H-Datei extrahiert. Damit kann man erreichen, daß Include-Dateien bei größeren Programmen nur noch die Header enthalten müssen, während der Code dazu bereits compiliert als .OBJ-Dateien in einer Bibliothek aufbewahrt wird. Damit verkürzen sich die Zyklen beim Entwickeln sehr, weil Compilierzeit gespart wird.

gen behandelt, damit alle, die keine NANs (von „Not a Number“) kennen und nicht die verschiedenen Rundungsmethoden, mehr über das Verhalten der Gleitkomma-Pakete und -Prozessoren wissen.



CAD und PC

Mit der Verbreitung der schnellen PC beginnt auch die Verbreitung von preiswerten und leistungsfähigen CAD-Programmen. Wir testen ein Paket, das für den elektronischen Bereich geeignet ist und sagen dazu einiges, was für CAD in der Praxis auch grundlegend sein sollte. Zum Beispiel sagt unser Tester: „Die Freiheiten, die ein System bietet, hat auch Nachteile. Die verantwortliche Handhabung des Systems liegt sehr stark beim Benutzer. Ohne Ausbildung kann er entsprechende Fehlhandlungen durchführen.“

Materialien zu mc

Zu mc gibt es die Sammeldisketten für PCs und Atari-Computer. Sie enthalten ausgewählte Programme aus den letzten Ausgaben. Rufen Sie 089/59 54 68 an, es meldet sich die Firma Shamrock, die in unserem Auftrag die Sammeldisketten für je 25 DM vertreibt. Fragen Sie nach dem Inhalt der Neuesten.



Neu von mc ist das PC-AT-Sonderheft. Es enthält kompakt alles, was man über den AT wissen muß. Hardware-Details werden am mc-modular-AT dargestellt. Im Sonderheft stehen viele Details über die Programmierung und die Software-Konfiguration eines ATs. Wissen, das man besitzen muß. Für 28 DM.



Das KI-Sonderheft von mc enthält kompaktes Wissen über das Gebiet der künstlichen Intelligenz. Lisp, Expertensysteme, KI-Programmiertechniken, Prolog, maschinelles Beweisen, Physik und Gehirn – das sind die wichtigsten Themen, die im Sonderheft zu finden sind. Es kostet 19.50 DM.



Know-how-Computer

Kostenlos bekommen Sie die Materialien zum berühmten Know-how-Computer, wenn Sie an den Verlag DM 2.50 in Briefmarken unter dem Stichwort know-how-Computer einsenden. Es gibt Klassensätze auf Anfrage.

**Ausgabe 9/88
erscheint am
29. August 1988**



**Bitte neueste Prospekte
anfordern!**



Auf alle Geräte 12 Monate Garantie. Preise gültig ab 1.12.87.
Lieferbedingungen auf Anfrage. MCI MICRO COMPUTER
INSTRUMENTS GMBH eingetragen AG Bergisch Gladbach
HRB 2575 · Herstellung und Vertrieb von Microcomputern.

MCI

Bensberger Straße 252 · 5060 Bergisch Gladbach 2
Tel. (02202) 1080
Fax: (02202) 31009 · Telex: 8873518

HIGHSCREEN

TOWER-POWER FÜR DEUTSCHLAND!

1799,-



279,-

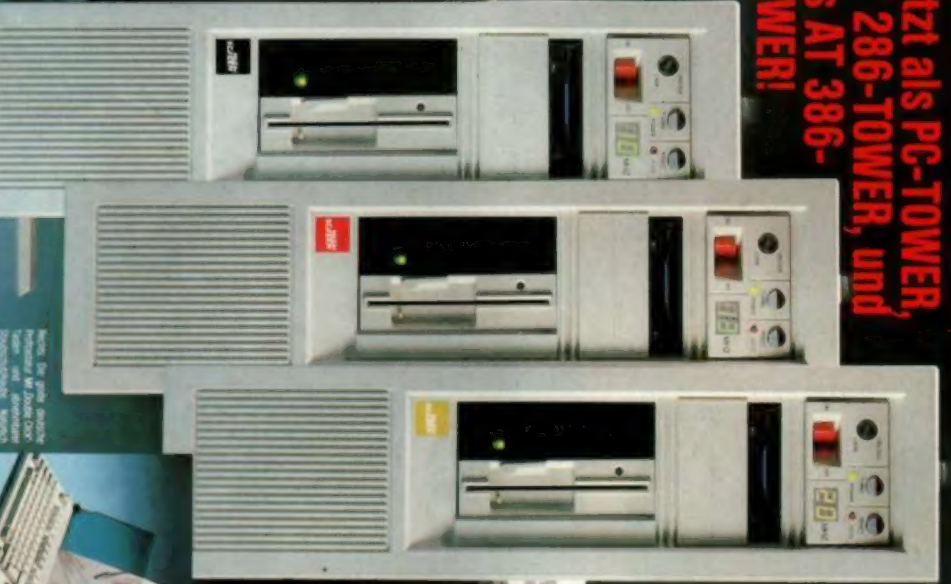
Jetzt als PC-TOWER, AT 286-TOWER, und als AT 386-TOWER!

HIGHSCREEN TOWER

Viel Platz im Computer und mehr Platz auf dem Schreibtisch. HIGHSCREEN-TOWER sind Profi-Computer, die kompetent und mit reichlich Platz für interne Erweiterungen. Und trotzdem stehen sie nicht. HIGHSCREEN-TOWER steht man neben dem Schreibtisch.

- Alle HIGHSCREEN-TOWER verfügen über eine professionelle Grundausstattung:**
- Schickes Voll-Metall-Gehäuse
 - Ungewöhnlich viel Platz für Erweiterungen
 - 2x 5,25" Halbhöhe Laufwerke bzw. Harddisks
 - 3x 3,5" Laufwerke 1" hoch oder 1 x 3,5" Laufwerk + 1 Harddisk 3,5"
 - 1 x 5,25" normal hoch
 - 1x 80 MB Harddisk
 - Mehr nach Landmarkt

- Dazu das nötige Netzteil 220 Watt
- Buch für den PC
- Große Tastatur mit abnehmbarer Schutzhaube
- Reiser-Schalter, Turbo-Schalter, Schlusssel-Schalter
- LED-Anzeige für Geschwindigkeit
- Mehr nach Landmarkt



Merke: Die große deutsche Hersteller-Welt. Die große deutsche Hersteller-Welt. Die große deutsche Hersteller-Welt.

HIGHSCREEN TOWER PC

- TOWER AT 286** 1799,-
- TOWER AT 386** 3195,-
- TOWER AT 486** 3699,-
- TOWER AT 586** 6499,-
- TOWER AT 686** 6999,-



798,-

3995,-

Modell	Prozessor	Arbeitsspeicher	Laufwerke	Preis
279	186	256	3x 5,25"	1799,-
3195	286	512	3x 5,25"	3195,-
3699	386	1024	3x 5,25"	3699,-
6499	486	2048	3x 5,25"	6499,-
6999	586	4096	3x 5,25"	6999,-



SE DESIGN AACHEN

LOBIS MICROCOMPUTER

kompetent und preiswert

- HAUPT-VERWALTUNG:** Postfach 1778, Roter Bruch 32-34, 5100 AACHEN, ☎ 0241/50 00 81, T 832 389 vobis d
- 1000 BERLIN 30** Kurtstienstr. 101 - 030/2 13 94 80
- 2000 HAMBURG** Kohnstam 15 - 040/2 79 46 76
- 2300 KIEL** Sophienblatt 74-78 - 0431/67 86 22
- 2800 BREMEN** Friedrich-Wilhelm-Str. 30 - 0203/2 78 63
- 4150 KREFELD** Ostwall 92 - 02151/80 07 93
- 3000 HANNOVER** Berliner Allee 47 - 0511/81 65 71
- 4000 DÜSSELDORF** Wielandstr. 21 - 0211/35 99 64
- 4100 DUISBURG 1** Friedrich-Wilhelm-Str. 30 - 0203/2 78 63
- 4300 ESSEN** Hüssensallee 3 - 0201/23 17 74
- 4600 DORTMUND** Hamburger Str. 110 - 0231/57 30 72
- 4800 BIELEFELD** Herforder Str. 106 - 0521/6 38 78
- 5000 KÖLN** Mathiasstr. 24-26 - 0221/24 86 42
- 5100 AACHEN** Viktoriastr. 74 - 0241/54 31 00
- 6000 FRANKFURT** Frankfurter Str. 207/209 - 069/73 40 49
- 7000 STUTTGART** Marienstr. 11-13 - 0711/60 63 36
- 6400 FULDA** Mittelstr. 19/21 - 0661/7 82 66
- 7500 KARLSRUHE** Kriegsstr. 27/29 (am BGH) - 0721/37 82 68
- 7750 KONSTANZ** Kreuzlinger Str. 18 - 07531/1 55 60
- 8000 MÜNCHEN** Abertstr. 3 - 089/77 21 10
- 8500 MÜNCHEN** Vordere Ledergrasse 8 - 0911/23 29 95
- 8900 AUGSBURG** Jakobstr. 16 - 0821/152349

Deutschlands umsatzgrößer Microcomputer-Spezialist

Fragen Sie die **LOBIS**-Fachberater in den Filialen oder über die **HOTLINE 07241 60741 60 40 11** Montag - Freitag 10-12 + 14-18 Uhr